PROCEEDINGS OF THE FOURTH INTERNATIONAL CONFERENCE ON SIMULATION OF ADAPTIVE BEHAVIOR

From animals to animats 4



DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

edited by Pattie Maes, Maja J. Mataric, Jean-Arcady Meyer,
Jordan Pollack, and Stewart W. Wilson

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
		Final 2/1/96 - 12/31/96

4. TITLE AND SUBTITLE
Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior

5. FUNDING NUMBERS
N00014-96-1-0475

6. AUTHOR(S)
Maes, Mataric, Meyer, Pollack, Wilson

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Brandeis University
Office of Sponsored Programs
415 South Street
Waltham, MA 02254

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Thomas McKenna
ONR 342CN
Office of Naval Research
800 N. Quincy Street
Arlington, VA 22217

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

19961021 191

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Collection of papers refereed and presented at the "SAB96" Conference.

DTIC QUALITY INSPECTED 1

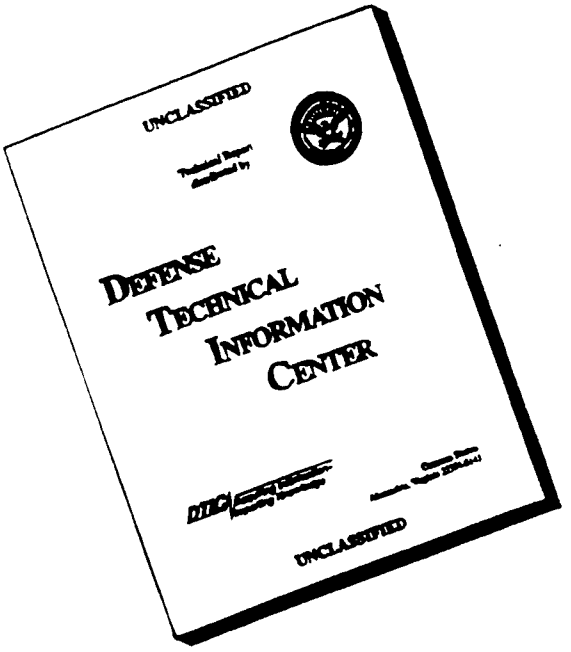
14. SUBJECT TERMS
AI, Robotics, Evolution, Ecology, Ethnology

15. NUMBER OF PAGES
644

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	

DISCLAIMER NOTICE



THIS DOCUMENT IS BEST QUALITY AVAILABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.

Complex Adaptive Systems

John H. Holland, Christopher G. Langton, and Stewart W. Wilson, advisors

Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence, John H. Holland

Toward a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life, edited by Francisco J. Varela and Paul Bourgine

Genetic Programming: On the Programming of Computers by Means of Natural Selection, John R. Koza

Genetic Programming: The Movie, John R. Koza and James P. Rice

From Animals to Animats 2: Proceedings of the Second International Conference on Simulation of Adaptive Behavior, edited by Jean-Arcady Meyer, Herbert L. Roitblat, and Stewart W. Wilson

Intelligent Behavior in Animals and Robots, David McFarland and Thomas Bösser

Advances in Genetic Programming, edited by Kenneth E. Kinnear, Jr.

Genetic Programming II: Automatic Discovery of Reusable Programs, John R. Koza

Genetic Programming II Video: The Next Generation, John R. Koza

Turtles, Termites, and Traffic Jams: Explorations in Massively Parallel Microworlds, Mitchel Resnick

From Animals to Animats 3: Proceedings of the Third International Conference on Simulation of Adaptive Behavior, edited by Dave Cliff, Philip Husbands, Jean-Arcady Meyer, and Stewart W. Wilson

Artificial Life IV: Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems, edited by Rodney A. Brooks and Pattie Maes

Comparative Approaches to Cognitive Science, edited by Herbert L. Roitblat and Jean-Arcady Meyer

Artificial Life: An Overview, edited by Christopher G. Langton

Evolutionary Programming IV: Proceedings of the Fourth Annual Conference on Evolutionary Programming, edited by John R. McDonnell, Robert G. Reynolds, and David B. Fogel

An Introduction to Genetic Algorithms, Melanie Mitchell

Catching Ourselves in the Act: Situated Activity, Interactive Emergence, and Human Thought, Horst Hendriks-Jansen

Toward a Science of Consciousness: The First Tucson Discussions and Debates, edited by Stuart R. Hameroff, Alfred W. Kaszniak, and Alwyn C. Scott

Genetic Programming: Proceedings of the First Annual Conference, edited by John R. Koza, David E. Goldberg, David B. Fogel, and Rick L. Riolo

Evolutionary Programming V: Proceedings of the Fifth Annual Conference on Evolutionary Programming, edited by Lawrence J. Fogel, Peter J. Angeline, and Thomas Bäck

Elements of Artificial Neural Networks, Kishan Mehrotra, Chilukuri K. Mohan, and Sanjay Ranka

From Animals to Animats 4: Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior, edited by Pattie Maes, Maja J. Mataric, Jean-Arcady Meyer, and Stewart W. Wilson

FROM ANIMALS TO ANIMATS 4

Proceedings of the Fourth International Conference
on Simulation of Adaptive Behavior

edited by
Pattie Maes, Maja J. Mataric, Jean-Arcady Meyer, Jordan Pollack,
and Stewart W. Wilson

A Bradford Book

The MIT Press
Cambridge, Massachusetts
London, England

© 1996 Massachusetts Institute of Technology

All rights reserved. No part of this book may be reproduced in any form by any electronic or mechanical means (including photocopying, recording, or information storage or retrieval) without permission in writing from the publisher.

This work relates to Department of Navy grant N00014-96-1-0475 issued by the Office of Naval Research. The United States government has a royalty-free license throughout the world in all copyrightable material contained within.

Printed and bound in the United States of America.

This publication may be ordered from The MIT Press, 55 Hayward Street, Cambridge, Massachusetts 02142 USA; by telephone: 1-800-356-0343 (toll-free) or 617-625-8569; by fax: 617-625-6660; by e-mail: mitpress-orders@mit.edu or www-mitpress.mit.edu.

ISSN 1089-4365
ISBN 0-262-63178-4

CONTENTS

Preface xi

THE ANIMAT APPROACH TO ADAPTIVE BEHAVIOR

Building "Fungus Eaters": Design Principles of Autonomous Agents 3
Rolf Pfeifer

Human Simulation of Adaptive Behavior:
Interactive Studies of Pursuit, Evasion, Courtship, Fighting, and Play 13
Philip W. Blythe, Geoffrey F. Miller, and Peter M. Todd

The Engineering of Mind 23
James S. Albus

PERCEPTION AND MOTOR CONTROL

Self-Taught Visually-Guided Pointing for a Humanoid Robot 35
Matthew Marjanovic, Brian Scassellati, and Matthew Williamson

A Self-Organizing Model of the Antennal Lobes 45
Rainer Malaka, Stefan Schmitz, and Wayne M. Getz

Some Adaptive Movements of Animats with Single Symmetrical Sensors 55
Owen Holland and Chris Melhuish

Categorization in a Real-World Agent Using Haptic Exploration and Active Perception 65
Christian Scheier and Dimitrios Lambrinos

How to Attract Females: Further Robotic Experiments in Cricket Phonotaxis 75
Barbara Webb and John Hallam

Coordination in a Six-Legged Walking System:
Simple Solutions to Complex Problems by Exploitation of Physical Properties 84
Holk Cruse, Christian Bartling, Jeffrey Dean, Thomas Kindermann, Josef Schmitz, Michael Schumm, and Hendrik Wagner

Orientation Behavior Using Registered Topographic Maps 94
Cynthia Ferrell

Locating Odor Sources in Turbulence with a Lobster Inspired Robot 104
Frank Grasso, Thomas Consi, David Mountain, and Jelle Atema

Dynamics for Vision-Based Autonomous Mobile Robots 113
Hartmut Neven, Axel Steinhage, Martin Giese, and Carsten Bruckhoff

Postural Primitives: Interactive Behavior for a Humanoid Robot Arm 124
Matthew M. Williamson

ACTION SELECTION AND BEHAVIORAL SEQUENCES

Action Selection Methods using Reinforcement Learning 135
Mark Humphrys

Towards Adaptive Behavior System Integration
Using Connectionist Infinite State Automata 145
Tom Ziemke

Centrally-Generated and Reflexive Control Strategies in the
Adaptive Behavior of Real and Simulated Animals 155
Jim H. Belanger and Mark A. Willis

Variable Binding and Predicate Representation in a Behavior-Based Architecture 163
Ian Horswill

The Experimental Study and Computer Simulation of Fish Behavior
in the Uniform Environment 173
V. A. Nepomnyashchikh and Vera A. Gremyatchikh

Handling Time-Warped Sequences with Neural Networks 180
Claudia Ulbricht

INTERNAL WORLD MODELS AND NAVIGATION

How Insects Learn about the Sun's Course: Alternative Modeling Approaches 193
Jeffrey Dickinson and Fred Dyer

Spatial Exploration, Map Learning, and Self-Positioning with MonaLysa 204
Jean-Yves Donnart and Jean-Arcady Meyer

Adaptive Animat Navigation Based on a Flexibility Model for the Environment 214
Peter Veelaert and Herbert Peremans

Maze Navigation Using Optical Flow 224
Andrew P. Duchon

An Autonomous System for Extracting Fuzzy Behavioral Rules in Mobile Robotics 233
A. G. Pipe and A. Winfield

MOTIVATION AND EMOTIONS

A New Control Architecture Combining
Reactivity, Planning, Deliberation and Motivation for Situated Autonomous Agent 245
François Michaud, Gérard Lachiver, and Chon Tam Le Dinh

A Finer-Grained Motivational Model of Behaviour Sequencing 255
Emmet Spier and David McFarland

What Are Emotions For?
Commitments Management and Regulation Within Animals/Animats Encounters 264
Michel Aubé and Alain Senteni

Reinforcement Learning and Animat Emotions 272
Ian Wright

LEARNING

Skinnerbots 285
David S. Touretzky and Lisa M. Saksida

No Bad Dogs: Ethological Lessons for Learning in Hamsterdam 295
Bruce M. Blumberg, Peter M. Todd, and Pattie Maes

Generalization in Instrumental Learning 305
Christian Balkenius

Learning to Use Selective Attention and Short-Term Memory in Sequential Tasks 315
Andrew Kachites McCallum

Explore/Exploit Strategies in Autonomy 325
Stewart W. Wilson

Learning Control Composition in a Complex Environment 333
E. G. Araujo and R. A. Grupen

Modular Growing Network Architectures for TD Learning 343
Pascal Blanchet

Learning to Detour & Schema-based Learning 353
Fernando J. Corbacho and Michael A. Arbib

Emergent Hierarchical Control Structures:
Learning Reactive/Hierarchical Relationships in Reinforcement Environments 363
Bruce L. Digney

A Model of Operant Conditioning for Adaptive Obstacle Avoidance 373
Paolo Gaudiano, Eduardo Zalama, Carolina Chang, and Juan López Coronado

Learning Navigational Behaviors Using a Predictive Sparse Distributed Memory 382
Rajesh P. N. Rao and Olac Fuentes

EVOLUTION

A Developmental Model for the Evolution of Complete Autonomous Agents 393
Frank Dellaert and Randall D. Beer

Evolution of Plastic Neurocontrollers for Situated Agents 402
Dario Floreano and Francesco Mondada

Increasing Adaptivity through Evolution Strategies 411
Ralf Salomon

Toward the Evolution of Dynamical Neural Networks for Minimally Cognitive Behavior 421
Randall D. Beer

Emergence of a Multi-Agent Architecture and New Tactics for the Ant Colony Food Foraging Problem Using Genetic Programming 430
Forrest H Bennett III

Cell Interactions as a Control Tool of Developmental Processes for Evolutionary Robotics 440
Peter Eggenberger

Evolution of the Sensorimotor Control in an Autonomous Agent 449
Susanne A. Huber, Hanspeter A. Mallot, and Heinrich H. Bülthoff

The Evolutionary Cost of Learning 458
Giles Mayley

Evolving Obstacle Avoidance Behavior in a Robot Arm 468
David E. Moriarty and Risto Miikkulainen

Automatic Generation of Adaptive Programs 476
Lee Spector and Kilian Stoffel

Sexual Swimmers: Emergent Morphology and Locomotion without a Fitness Function 484
Jeffrey Ventrella

COEVOLUTION

Cooperative Versus Competitive System Elements in Coevolutionary Systems 497
Robert E. Smith and H. B. Cribbs III

Co-evolution of Pursuit and Evasion II: Simulation Methods and Results 506
Dave Cliff and Geoffrey F. Miller

Incremental Self-Improvement for Life-Time Multi-Agent Reinforcement Learning 516
Jieyu Zhao and Jürgen Schmidhuber

Dynamics of Co-evolutionary Learning 526
Hugues Juillé and Jordan B. Pollack

COLLECTIVE BEHAVIOR

Oscillation-Enhanced Adaptability in the Vicinity of a Bifurcation: The Example of Foraging in Ants 537
Eric Bonabeau and François Cogne

Dominance Interactions, Spatial Dynamics and Emergent Reciprocity in a Virtual World 545
Charlotte K. Hemelrijk

A Study of Territoriality: The Role of Critical Mass in Adaptive Task Division 553
Miguel Schneider Fontán and Maja J Mataric

Emergent Adaptive Lexicons 562
Luc Steels

Synthetic Robotic Language Acquisition by Observation 568
Alexandros Moukas and Gillian Hayes

The Evolution of Communication Schemes over Continuous Channels 580
Gregory M. Saunders and Jordan B. Pollack

Using A-Life to Study Bee Life: The Economics of Central Place Foraging 590
P. de Bourcier

An Evolved Fuzzy Reactive Control System for Co-operating Autonomous Robots 599
Robert Ghanea-Hercock and David P Barnes

On Simulating the Evolution of Communication 608
Jason Noble and Dave Cliff

Collective Behavior by Modular Reinforcement-Learning Animats 618
Norihiko Ono, Kenji Fukumoto, and Osamu Ikeda

Robotic "Food" Chains: Externalization of State and Program for Minimal-Agent Foraging 625
Barry Brian Werger and Maja J Mataric

(Not) Evolving Collective Behaviours in Synthetic Fish 635
Nahum Zaera, Dave Cliff, and Janet Bruten

Author Index 645

PREFACE

The papers in this book were presented at the Fourth International Conference on Simulation of Adaptive Behavior (SAB96), held at North Falmouth, Cape Cod, Massachusetts, USA, on September 9-13, 1996. The objective of the biennial SAB conferences is to bring together researchers from a wide range of backgrounds including ethology, theoretical biology, psychology, artificial life, machine learning and robotics, to further our understanding of the behaviors and underlying mechanisms that allow natural and artificial animals ('animats') to adapt and survive in uncertain environments.

Adaptive Behavior research is distinguished by its focus on the modeling and creation of complete animal-like systems, which—however simple at the moment—may be one of the best routes to understanding intelligence in natural and artificial systems. The field received initial recognition on the occasion of the first SAB conference, which was held in Paris in September, 1990. Subsequent SAB conferences (Hawaii, December 1992 and Brighton, England, August 1994) drew increasing numbers of papers and participants. In 1992, The MIT Press introduced the quarterly journal *Adaptive Behavior*, and The International Society for Adaptive Behavior (ISAB) was established in 1995—both events further marking the emergence of Adaptive Behavior as a full-fledged scientific discipline. The present proceedings are a comprehensive and up-to-date resource for the latest progress in this exciting field.

The 66 papers presented at the conference and published here were selected from 150 submissions after a two-pass review process designed to ensure high and consistent overall quality. The authors focus on well-defined models, computer simulations, and robotics demonstrations that help characterize and compare various organizational principles or architectures capable of inducing adaptive behavior in real animals or synthetic agents. The papers are ordered according to the scale at which adaptive behavior takes place, ranging from immediate adaptation in sensorimotor control, to learning within an animat's lifetime, to adaptive behavior exhibited by successive generations of animats, and finally to adaptive collective behavior of animats in groups.

In addition to the presentations of accepted papers, a number of distinguished researchers gave keynote lectures. The SAB96 Keynote Lecturers were:

James S. Albus, Head of the Intelligent Systems Division at the National Institute of Standards and Technology, pioneer in robotics and neural networks, and author of *Brains, Behavior, and Robotics*, whose paper "The Engineering of Mind" is included in this volume;

Jelle Atema, Professor of Biology at Boston University and the director of the Boston University Marine Program at Woods Hole Marine Biology Laboratory;

Daniel Dennett, Distinguished Arts and Sciences Professor and Director of the Center for Cognitive Studies at Tufts University, the author of *Consciousness Explained* and *Darwin's Dangerous Idea*;

C. R. Gallistel, Professor of Psychology at UCLA and the author of *The Organization of Learning* and *Animal Cognition*;

J. A. Scott Kelso, Director of the Center for Complex Systems at Florida Atlantic University and author of *Dynamic Patterns: The Self-Organization of Brain and Behavior*;

David Touretzky, Senior Research Scientist in the Computer Science Department and the Center for the Neural Basis of Cognition at Carnegie Mellon University, author of *The Mathematics of Inheritance Systems*.

The conference and its proceedings could not have happened without the substantial help of a wide range of people. First and foremost, we would like to thank the members of the Program Committee. They thoughtfully reviewed all the submissions and worked with many of the authors to ensure a high standard in the accepted papers. The Committee members were:

Peter Angeline, USA
Randall Beer, USA
Lashon Booker, USA
Thomas Collett, UK
Jacques Ferber, France
Simon Giszter, USA
Inman Harvey, UK
Phil Husbands, UK
Harry Klopf, USA
David McFarland, UK
Geoffrey Miller, Germany
Alan Schultz, USA
Tim Smithers, Spain
Luc Steels, Belgium
Frederick Toates, UK
Saburo Tsuji, Japan
David Waltz, USA
Ronald Arkin, USA
Bruce Blumberg, USA
Dave Cliff, UK
Holk Cruse, Germany
Dario Floreano, Italy
John Hallam, UK
Ian Horswill, USA
Leslie Pack Kaelbing, USA
Michael Littman, USA
José del R. Millán, Italy
Rolf Pfeifer, Switzerland
Jean-Jacques Slotine, USA
Emmet Spier, UK
Lynn Andrea Stein, USA
Peter Todd, Germany
Patrick Tufts, USA

Professor Herbert Roitblat of the University of Hawaii contributed his wisdom and experience with previous SAB conferences throughout the planning and preparation for SAB96. We are very grateful to him.

We are indebted to our sponsors,

The Office of Naval Research for generous financial support;
The M. R. Bauer Foundation for its support of academic activities at the Volen Center for Complex Systems of Brandeis University;
The International Society for Adaptive Behavior for early support and sponsorship.

We are grateful to Dr. Thomas McKenna of ONR for his advice and guidance.

The enthusiasm and hard work of numerous individuals was essential to the conference's success. We thank Christie Davidson of the MIT Media Lab for administration of the program and review process. We are grateful to the Brandeis Computer Science Department, especially Myrna Fox, Jeanne DeBaie, and the student staff of the Department office, for their work in the local, promotional, and financial aspects of the meeting. Alan Danziger, Jeremy Gilbert, and Aryeh Primus provided indispensable World-Wide-Web and computer support.

Finally, we are once again especially indebted to Jean Solé for the artistic conception of the SAB96 poster and the proceedings cover.

We invite readers to enjoy and profit from the papers in this book, and look forward to the next conference, SAB98.

Pattie Maes, Program Chair

Maja Mataric Jean-Arcady Meyer Jordan Pollack Stewart W. Wilson

THE ANIMAT APPROACH TO ADAPTIVE BEHAVIOR

Building "Fungus Eaters": Design Principles of Autonomous Agents

Rolf Pfeifer
AI Lab, Computer Science Department
University of Zurich, Winterthurerstrasse 190
CH-8057, Zurich, Switzerland
pfeifer@ifi.unizh.ch

Abstract

We describe a set of design principles for building "Fungus Eaters". "Fungus Eaters" are complete autonomous systems. The goal is to extract and describe in a compact way a large part of the insights which have been acquired in the animats field. The principles have been developed from a cognitive science perspective. Although they represent only a very modest beginning, they make immediately clear what sort of ideas about intelligence and cognition they endorse. They all contrast sharply with classical thinking. Moreover, they provide powerful heuristics for design.

1 Introduction

In their review paper of the first SAB conference in 1990, Jean-Arcady Meyer and Agnès Guillot argue that the animat approach will play an important role in resolving some of the fundamental controversies in the study of intelligence or cognition (Meyer and Guillot, 1991). Four years later, at the third SAB conference, they propose three types of goals for animat research, short term, intermediate term, and ultimate goals. In the short term it is the discovery and exploration "... of architectures and working principles that allow a real animal, a simulated animal, or a robot to exhibit a behavior that solves a specific problem of adaptation in a specific environment." (Meyer and Guillot, 1994, p. 7). In the intermediate term, it is the generalization of this knowledge in order to better understand the relation between architectures, working principles and adaptive performance vis-à-vis different types of environments. The ultimate goal, then, is to understand the adaptive value and working principles of human cognition. They conclude by stating that "... the domain is in definite need of theoretical advances that could provide useful generalizations of still highly disparate pieces of knowledge" (p. 8). This paper is an attempt to make a—however modest—contribution towards generalization. The contribution will be in the form of a set of design principles of autonomous agents.

Currently, there is no generally accepted theoretical framework. Although there have been some efforts at developing overarching theories, they are typically only recognized and taken up by a small part of the community. Examples are the "Behavioral Economics" approach (McFarland and Bösser, 1993), the dynamical systems approach (Beer, in press; Steinhage and Schöner, in press), and the evolutionary approach (for a review, see Harvey et al., in press). Maes, in a review paper, tries to capture some general principles contrasting the traditional and the animat approach (Maes, 1992).

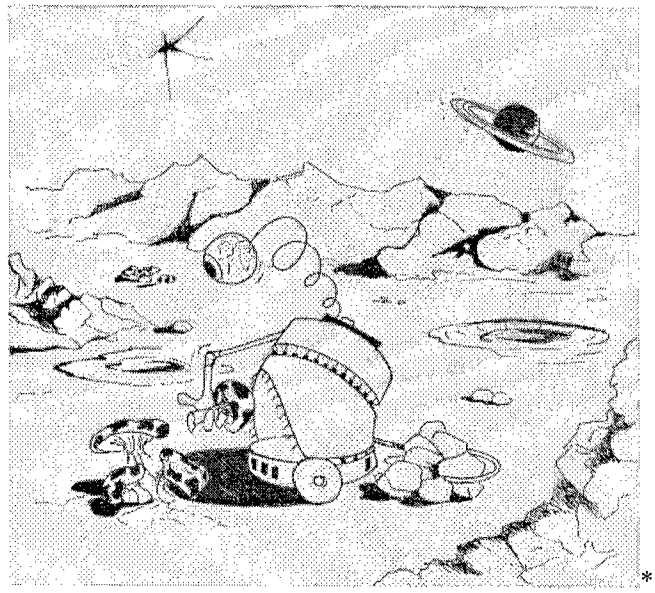


Figure 1: A "Fungus Eater" ingesting fungus on a distant planet. It has to perform its task autonomously while maintaining its energy supply (Cartoon by Isabelle Follath, Zurich).

In this paper we will focus on "Fungus Eaters" or real-world autonomous agents. We do not further discuss work that involves simulation only. "Fungus Eaters" are a particular species of animats. The term is inspired by Masanao Toda's seminal book entitled "Man, Robot, and Society" (Toda, 1982). Briefly, "Fungus Eaters" are complete autonomous creatures, sent to a distant planet for collecting uranium ore. They have to worry about energy supply—they feed on a particular type of fungus that grows on the planet—and predators, while performing their task (figure 1). Toda suggested the study of "Fungus Eaters" out of a dissatisfaction with the way psychology, in particular cognitive psychology, was going at the time. The main point was that we should study "complete" systems, however

simple, rather than only isolated faculties like planning, memory, or decision making.

One of the reasons for the lack of consensus is that the field is very new. Another one is that what we consider to be a good design of an autonomous agent or an interesting and valuable theory, depends on the goals we have in mind. If we want to build robots that collect garbage we are looking for something very different than the biologist who is trying to understand evolution. Or the computer scientist who is interested in the power of evolutionary algorithms is after something else than the psychologist trying to understand cognition. The interdisciplinary nature of the field adds to its diversity.

Although this diversity bears a lot of creative potential, it might nevertheless be useful to try and ferret out some of the accumulated insight and represent it in a compact form. We have tried to capture some of the results as a set of "design principles of autonomous agents". The principles presented here have emerged out of five years of intensive research on various aspects of the animat field, or "New AI", and prior to that over 10 years of research in traditional AI and psychological modeling (e.g. Pfeifer, 1988; Pfeifer, 1994; Pfeifer, 1995; Pfeifer et al., 1989; Pfeifer and Verschure, 1992, 1995). They do not constitute a "theory", but they could represent a starting point for discussion. Ultimately, the idea would be to discover the "theory" from which these principles can be derived. We put "theory" in quotes to indicate that it is an entirely open question whether there will ever be one unifying theory of intelligence or cognition. We are not saying that everyone should agree with these principles. But we do hope that they will help to focus a debate about the underlying principles of naturally intelligent systems.

We begin with a short argument of why we chose the form of "design principles". We then present some reflections on the design process from a cognitive science and an engineering perspective. Then we describe a set of design principles. We conclude with some comments on what we have achieved and what should be done next.

2 Why "Design principles"?

The short answer to this question is that the design perspective is highly productive. The animat approach is by definition synthetic. The underlying slogan is "understanding by building". Design principles provide guidance on how to build animats. The way we build our animats is a manifestation of our views of intelligence. One purpose of the design principles is to make this knowledge explicit. The great advantage of the synthetic approach is, of course, that we have built the agents ourselves, i.e. we know what is in our systems, and that we can experiment with alternatives as much as we like. This experimental freedom also accounts for the popularity of computer simulation models.

In this paper we do not want to study simulation but "Fungus Eaters". "Fungus Eaters" are "complete" in the sense that everything needed for behaving in the real world has to be there. It is not sufficient to model only one aspect, say its memory or its perceptual system. On the one hand, this makes it harder, but on the other, it constitutes the real power of the approach .

There is an additional point that makes the design perspective especially attractive. Natural animats, i.e. animals, can be productively viewed from a designer's perspective: evolution as a designer, perhaps a blind and slow one, but nevertheless a designer, and a good one at that (e.g. Dawkins, 1986). McFarland's "animal robotics" approach capitalizes on this point (e.g. McFarland and Bösser, 1993).

Before discussing the design principles, let us briefly look at some issues in design.

3 Engineering and cognitive science

Assume that the task is to build a robot that collects ping-pong balls in a particular room as quickly as possible. Figure 2 illustrates two alternative designs. The solution on the left shows a powerful vacuum cleaner, sucking in the ping-pong balls at great speed. On the right, there is a robot with sensors and manipulators, and with mechanisms that enable the robot to learn distinctions between different kinds of objects and to learn grasping and carrying light, delicate objects without hurting them. From an engineering perspective, the robot on the left is perfect. The only considerations are performance and price. The performance criterion in this case is obvious, namely the number of balls collected per unit time. It turns out to be much harder to evaluate the performance of autonomous agents. There are promising first attempts (e.g. Gat, in press; Hemelrijk and Lambrinos, 1994; Mataric, 1995; Smithers, 1995), but there is no consensus. Now, the design principles may also be used to assess whether a particular design is of potential interest from a cognitive science point of view.

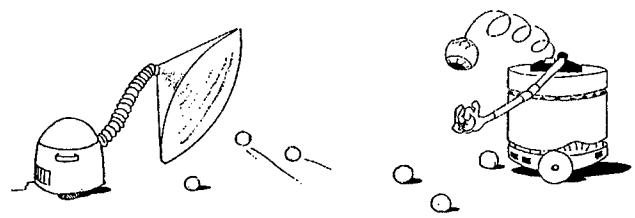


Figure 2: One the left the engineering solution, on the right, the cognitive science solution.

In a cognitive science context, what matters is of a completely different nature than what is relevant for engineering. While performance is certainly a criterion, it is by no means the only one, nor is it the most important one. In cognitive science the important question is what we can

learn about intelligence from our robot. And it seems that the robot on the right in figure 2 can teach us more. The kinds of behaviors it can display are more interesting (flexibility and adaptivity). From the robot on the left we can learn about good engineering, but only little about cognitive science.

4 Design principles of autonomous agents

4.1 Types of explanations

There is a kind of "meta principle" that has to be endorsed if the design principles are to make sense. The meta principle suggests that designs for "Fungus Eaters" always be evaluated from three different perspectives, namely functional, learning and development, and evolutionary. Since our goal is to understand intelligence, we should always keep these three types of explanations in mind. Experience has shown that they contribute in complementary ways to our understanding. While we may choose to focus on one of them we should demonstrate compatibility with the others.

The *functional* perspective[1] explains why a particular behavior is displayed by an agent based on its current internal and sensory state, given its physical set-up. Often, this kind of explanation is used in engineering. But also in cognitive science it is highly productive. Just think of the creative nature of the Braitenberg vehicles, where it is surprising how seemingly sophisticated kinds of behavior result from very simple mechanisms (Braitenberg, 1984).

The *learning and developmental* perspectives not only resort to the current internal state, but to some events in the past in order to explain the current behavior. They provide explanations of how the actual behavior came about. The distinction between learning and development is that development includes maturation of the organism, whereas learning does not.

Evolutionary explanations put the agent into the context of an evolutionary process. The fact that we argue with evolutionary principles does not mean that we have to reproduce evolution in simulation. Biologists have talked for many years about evolution without making simulation models. We have included evolutionary consideration throughout the paper, but we do not specifically elaborate the design principles underlying simulated evolution. For an excellent review, see Harvey et al. (in press).

There is a somewhat orthogonal perspective on intelligent systems, namely the one of societies of agents. A satisfactory explanation of intelligence would also have to include aspects of social systems. While at the functional level, society is not an issue, in ontogenetic and phylogenetic development, it is an essential perspective. We will not go further into this aspect in this paper.

[1]The term "functional" is used in different ways in the literature. Here the term is used to distinguish one level of explanation from a learning/developmental and an evolutionary one.

4.2 Classes of principles

There are three classes of design principles. The first one concerns the kinds of agents and behaviors that are of interest from a cognitive science perspective. The second concerns the agent itself, its morphology, its sensors and effectors, its control architecture, and its internal mechanisms. The third class contains principles that have to do with ways of thinking and proceeding, with stances, attitudes, and strategies to be adopted in the design process. Because of space limitations we will focus on the first and the second class, and only briefly mention the third. An overview of the principles is given in table 1.

Table 1: Summary of design principles

Principle	Name
Types of agents of interest, ecological niche and tasks	
1	The "complete agents" principle
2	The "ecological niche" principle
Morphology, architecture, mechanism	
3	The principle of parallel, loosely coupled processes (the "anti-homunculus" principle)
4	The "value" principle
5	The principle of sensory-motor coordination
6	The principle of "ecological balance"
7	The principle of "cheap designs"
Strategies, heuristics, stances, metaphors	
8	"Frane-of-reference" principle
9	"Constraints" principles
10	Compliance with principles
	etc.

4.3 Type of agents, ecological niche, and tasks

Principle 1: The "Fungus Eaters" principle

As pointed out above, the kinds of agents of interest are the "Fungus Eaters". They are "complete systems", i.e. systems capable of performing a set of tasks in the real world independently and without human intervention. In other words, they are (a) autonomous, (b) self-sufficient, (c) embodied, and (d) situated.

Principle 1a: The agents must be *autonomous*, i.e. they have to be able to function without human intervention, supervision, or instruction.

Principle 1b: The agents must be *self-sufficient*, i.e. they have to be able to sustain themselves over extended periods of time. They have to be able to perform a set of tasks, including maintaining themselves (keeping a sufficient

energy level, keeping clean, lubricated, undamaged, etc.), without incurring an irrecoverable deficit in any of its resources. This principle imposes constraints on the architecture (see below).

Principle 1c: The agents must be *embodied*, i.e. they must be realized as a physical system capable of acting in the real world. Although simulation studies can be extremely helpful in designing agents, building them physically typically leads to surprising new insights. This point has been forcefully made by Brooks (1991). Physical realization often facilitates solutions which might seem hard if considered only in an information processing context. An agent existing only in simulation would not be complete.

Principle 1d: The agents must be *situated*, i.e. the whole interaction with the environment must be controlled by the agent itself, i.e. the world must always be seen from the perspective of the agent. Moreover, the agent has to be able to bring in its own experience in dealing with the current situation.

The "Fungus Eater" perspective implies that the agent be studied over extended periods of time. This time span is relevant because we are specifically interested in how agents evolve over time, either on an ontogenetic time scale (which includes learning), or an evolutionary one. In physical agents there is at best a learning perspective—developmental and evolutionary ones are confined to simulation because of technological problems.

An example of what a "Fungus Eater" might look like is shown in the cartoon in figure 1. True "Fungus Eaters" that fulfill all the criteria of principle 1 still do not exist.

Note the contrast to classical views of intelligence where often only performance on one particular problem solving task was at issue.

References supporting this principle include: Brooks, 1991; McFarland and Bösser, 1993; Toda, 1982.

Principle 2: The principle of the ecological niche

There is no universality in the real world. Animats are always designed for a particular niche. The concepts of autonomy, self-sufficiency, deficits, etc. only make sense with respect to an ecological niche. This implies defining all the tasks the agent has to fulfill. Note that the definition of the task is, in a sense, independent of the agent itself. The designer decides what the tasks of the agent are and he will design it such that it will accomplish them. This does not mean that there must be an explicit representation of the task within the agent. This point is nicely illustrated by one of Maja Mataric's remarks about the behavior of her robots: "They're flocking, but that's not what they think they are doing" (quoted in Dennett, in press).

The description of the ecological niche also includes the kinds of possible competition (e.g. McFarland, 1991). A garbage collecting robot has to compete with other robots, but also with other machines and with humans. Moreover, environments may be characterized formally. A well-known example is the characterization in terms of so-called sensory-state machines, as class 0, 1 or 2 environments (Wilson, 1991).

The definition of the ecological niche provides useful constraints for the design of the agent. An illustrative example is Ian Horsewill's robot Polly. It is based on a cheap vision system that exploits the fact that office floors are flat. If the floors are flat, a higher y-coordinate implies that the object is further away (given the object is standing on the ground). This is illustrated in figure 3. In addition, learning problems that are intractable if considered from a purely computational view, often turn out to be benign, if the constraints of a particular ecological niche are taken into account (see below, principle 7).

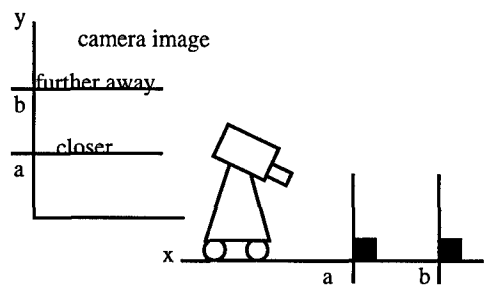


Figure 3: Ian Horsewill's robot, Polly, exploiting constraints of an ecological niche (flat floors in offices).

Evaluation of agents always has to be done with respect to a particular ecological niche, otherwise a comparison of performance is not possible.

Classical views of intelligence do not include the notion of an econiche because programs are restricted to virtual spaces.

References supporting this principle include: Horsewill, 1992; McFarland, 1991; McFarland and Bösser, 1993.

4.4 Morphology, architecture, mechanism

Principle 3: The principle of parallel, loosely coupled processes

In essence, this principle states that intelligence or cognition is emergent from a large number of parallel, loosely coupled processes. These processes run asynchronously and are largely peripheral, requiring little or no centralized resources. Principle 3 could also be called the "anti-homunculus" principle. It is directly motivated from biology.

A strong proponent of this principle is Brooks (e.g. 1991). The Braitenberg vehicles (Braitenberg, 1984), the extended Braitenberg architectures (e.g. Scheier and Pfeifer, 1995), Action Selection Dynamics (Maes, 1991), and PDL (Steels, 1992) can be seen in the same spirit. In all of these approaches, there is no "faculty" deciding on what to do next, i.e. there is no centralized action selection mechanism.

One of the main claims made here is that coherent behavior can be achieved without central control. A beautiful example that fully endorses this principle is the Cog project (Brooks, 1994; Brooks and Stein, 1993). In our own work we have applied this principle in all our agents. They employ an Extended Braitenberg Architecture (EBA), a straightforward generalization of standard Braitenberg architectures (e.g. Lambrinos, 1995; Scheier and Pfeifer, 1995).

While this principle is accepted by many researchers where lower levels of intelligence (e.g. insects, reptiles) are concerned, it is often contested when applied to human cognition. We feel that the principle should be maintained much longer and not given up until there is unequivocal evidence for the need of "higher" processes (Pfeifer, 1995).

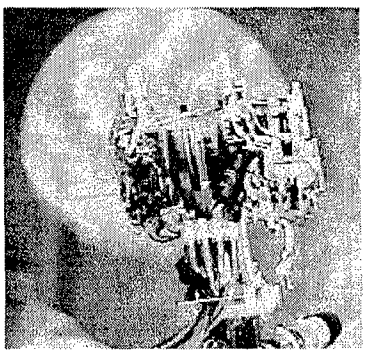


Figure 4: Cog—a robot displaying coherent behavior based on many parallel, loosely coupled processes.

The principle of parallel, loosely coupled processes contrasts sharply with classical thinking where a centralized seat of intelligence is assumed. Classical thinking does not object to parallel processes. The objection is that coherence cannot be achieved unless there is central integration.

References supporting this principle include: Braitenberg, 1984; Brooks, 1991; Brooks and Stein, 1993; Maes, 1991; Steels, 1992; Scheier and Pfeifer, 1995; Pfeifer and Scheier, in press

Principle 4: The "value" principle

This principle states that the agent has to be embedded in a value system, and that it must be based on self-supervised learning mechanisms employing principles of self-organization. If the agent is to be autonomous and situated it has to have a means to judge what is good for it and what isn't. This is achieved by a value system, a fundamental aspect of every "Fungus Eater" and more generally of every animat.

There is an implicit and an explicit aspect of the value system. In a sense, the whole set-up of the agent constitutes value: the designer decides that it is good for the agent to have a certain kind of locomotion (e.g. wheels), certain sensors (e.g. IR sensors), certain reflexes (e.g. turn away from objects), certain learning mechanisms (e.g. selectionist learning), etc. These values are implicit. They are not represented explicitly in the system. To illustrate the point, let us look at reflexes for a moment. Assume that a garbage collecting robot has the task to collect only small pegs and not large ones. Moreover, it should learn this distinction from its own perspective. The agent is equipped with a number of reflexes: turning away from objects, turning towards an object, and grasping if there has been lateral sensory stimulation over a certain period of time. The value of the first reflex is that the agent should not get damaged. The second and the third reflex increase the probability of an interesting interaction. Note that this interpretation in terms of value is only in the eye of the designer—the agent will simply execute the reflexes.

These reflexes introduce a bias. The purpose of this bias is to speed up the learning process because learning only takes place if a behavior is successful. If the behavior is successful, i.e. if the agent manages to pick up a peg, a value signal is generated. In this case, an *explicit* value system is required. In this way, the intuition that grasping is considered rewarding in itself, can be modeled. Figure 5 shows a learning robot, receiving a value signal because it has successfully grasped an object.

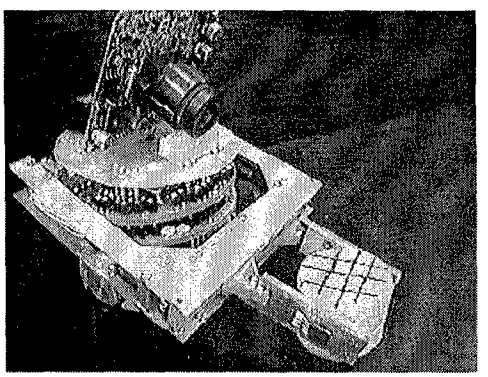


Figure 5: The garbage collecting robot has succeeded in grasping a small peg. An explicit reinforcement signal is generated which enables the robot to eventually learn the distinction between small (graspable) and large (non-graspable) pegs.

According to the "value" principle, the learning mechanisms have to be based on principles of self-organization, since the categories to be formed are not known to the agent beforehand. Examples are competitive schemes (e.g. Kohonen, 1988; Martinetz, 1994), or selectionist ones (Edelman, 1987).

If we were only interested in performance, there would be easier solutions. But the processes described are fundamental for understanding cognition. There is increasing evidence that categorization and concept formation in human infants is strongly based on value systems and processes of self-organization (Thelen and Smith, 1994).

This view of value systems and self-organization contrasts with classical thinking. The metaphor of information processing that underlies traditional AI and cognitive science, cannot accommodate self-organization. The "value" principle is closely related to the principle of sensory-motor coordination and ecological balance.

References supporting this principle include: Edelman, 1987; Pfeifer and Verschure, 1992; Pfeifer and Scheier, in press; Thelen and Smith, 1994;

Principle 5: The principle of sensory-motor coordination

This principle states that the interaction with the environment is to be conceived as a sensory-motor coordination. Sensory-motor coordination involves the sensors, the control architecture, the effectors, and the agent as a whole. A consequence of this principle is that classification, perception, and memory should be viewed as sensory-motor coordinations rather than as individual modules (e.g. Dewey, 1896; Douglas, 1993; Edelman, 1987).

Normally, perception is viewed as a process of mapping a proximal (sensory) stimulus onto some kind of internal representation. The enormous difficulties of classical computer vision to come to grips with the problem of invariances suggests that there may be some fundamental problems involved. Viewing perception as sensory-motor coordination has a number of important consequences.

From an information theoretic view, the sensory-motor coordination leads to a dimensionality reduction of the high-dimensional sensory-motor space (Pfeifer and Scheier, in press). This reduction allows learning to take place even if the agent moves. In fact, movement itself is beneficial since through its own movement, the agent *generates* correlations in the interaction with the environment. The second important aspect of sensory-motor coordination is the generation of cross-modal associations, including proprioceptive cues originating from the motor system (Thelen and Smith, 1984; Scheier and Lambrinos, 1996).

Additional support for the principle of sensory-motor coordination comes from developmental studies. There is a lot of evidence that concept formation in human infants is directly based on sensory-motor coordination (Thelen and Smith, 1984; Smith and Thelen, 1993; see figure 6). The concepts of humans are thus automatically "grounded". Similarly, if this principle is applied to artificial agents, the latter will only form fully grounded categories. The symbol grounding problem is really not an issue—anything the agent does will be grounded in its sensory-motor coordination. Note that the terms categorization and concept building are entirely observer-based. They relate only to the behavior of the infant, not to any sort of internal mechanism.

There is another kind of approach that closely relates to this principle, namely active vision (e.g. Ballard, 1991). Vision is not seen as something that concerns only input, but movement is considered to be an integral aspect.



Figure 6: Infant categorizing objects and building up concepts while engaged in sensory-motor coordination.

As already alluded to, this view contrasts with the traditional view of perception as a process of mapping a proximal stimulus onto an internal representation. In the view proposed here, the object representation is in the sensory-motor coordination. "Recognizing" an object implies re-enacting a sensory-motor coordination. Most objections to this view of perception have their basis in introspection. The latter has long ago been demonstrated to be a poor guide to research (Nisbett and Wilson, 1977).

References supporting this principle include: Ballard, 1991; Dewey, 1896; Douglas, 1993; Edelman, 1987; Thelen and Smith, 1994; Smith and Thelen, 1993; Scheier and Lambrinos, 1996; Pfeifer and Scheier, in press; Scheier and Pfeifer, 1995;

Principle 6: The principle of "ecological balance"

The principle of "ecological balance" states that there has to be a match between the "complexity" of the sensors, the actuators, and the neural substrate. Moreover, it states that the tasks have to be "ecologically" adequate. The way the term "complexity" is used here, appeals to our everyday understanding: a human hand is more complex than a forklift, a CCD camera more complex than an IR sensor.

From this principle we can get considerable leverage. Let us look at an example illustrating how *not* to proceed. Assume that we have a robot with two motors and a few IR sensors, say the robot Khepera™. In some sense, this design is balanced due to the intuition of the engineers that built it (except that its processor is too powerful if it is fully exploited). Assume further that some researchers have become frustrated because with the IRs they can only do very simple experiments. They would like to do more interesting things like landmark navigation.

The next logical step for them is to add a CCD-camera. It has many more dimensions than the few IR sensors. The rich information from the camera is transmitted to a central device where it is processed. This processing can, for example, consist in extracting categories. But the categories are formed as a consequence of a sensory-motor coordination. Because the motor system of the agent is still the same, the resulting categories will not be much more

interesting than before (although they may be somewhat different). Trying to build categories using only the visual stimulation from the camera (not as a sensory-motor coordination) would violate principle 5. Classical computer vision has violated this principle—and the problems are well-known. It would be a different story if, together with the CCD camera, additional motor capabilities would have been added to the robot, like a gripper or an arm of sorts. Figure 7 shows a balanced design on the left , an unbalanced one in the middle, and again a more balanced one on the right.

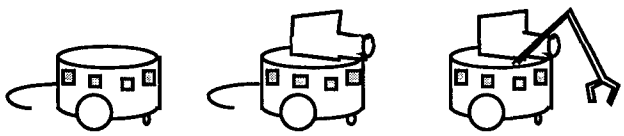


Figure 7: Balanced design on the left, unbalanced design in the middle, and again more balanced design on the right.

An approach that is fully compatible with the principle of "ecological balance" is again the Cog project. More sophistication on the sensor side (two eyes, each with a camera for peripheral and foveal vision), is balanced by more complexity on the motor side. The arm and the hand are quite sophisticated. Moreover, the head and the eyes can all move which leads to a system of a very large number of degrees of freedom. A lot of the processing is done peripherally, and the central processing capability is not inflated artificially. It is not surprising that Cog fulfills this design principle. It was Brooks who pointed out that tasks need to be ecologically appropriate (Brooks, 1990). In particular he argued that "elephants don't play chess." We couldn't agree more.

Important evidence for this principle comes also from studies in infant psychology by Bushnell and Boudreau (1993). Their results suggests that there is in fact a kind of co-evolution in the sensory-motor development of the infant. Roughly speaking, acuity of visual distinctions highly correlates with precision of motor movement.

Again, this view sharply contrasts with traditional AI and cognitive science, where intelligence was seen as centralized information processing, with no, or very little consideration given to the physical set-up. A concept like "ecological balance" would not make sense in that framework.

References supporting this principle include: Brooks, 1991, 1994; Pfeifer, 1995; Smith and Thelen, 1993; Bushnell and Boudreau, 1993.

Principle 7: The principle of "cheap design"

The principle of "cheap design" states that good designs are "cheap". This requires a bit of explanation. "Cheap", as used here, includes various components.

First, cheap means literally cheap. If the robot is built cheaply, we have to worry about physics. For example, the well-known Puma™ arm is not cheap. It is, in a way, "too good": it can be programmed without having to worry much about the real world. In a sense, the real world has largely been taken care of by the engineers. The programmer can simply choose the angles, and the arm will move to the requested position, as long as it is physically possible. If the arm is worse, if it is not so neatly engineered, the programmer has to worry much more about forces, about eigenfrequencies, about sensory-motor coordination, about interacting with the environment. Incorporating considerations about the physics into designs, typically leads to better and more robust designs. In this sense, cheap means capitalizing on the system-environment interaction.

A lovely illustration of exploitation of the physics is insect walking. Leg coordination in insects does not require a central controller. There is no internal process corresponding to global communication between the legs, they communicate only locally with each other (e.g. Cruse, 1991). But there *is* global communication between all the legs, namely through the environment. If the insect lifts one leg, the force on all other legs is changed instantaneously because of the weight of the insect. This communication is exploited for the purpose of coordination.

Second, cheap means parsimonious in the traditional sense of Occam's razor. If there are several models or designs achieving the same task performance, the most parsimonious model is considered the best. Even if the term "parsimonious" is subject to debate, the general idea is clear and generally accepted as a scientific principle. But of course, this depends on what we consider to be the task and the environment (principles 1 and 2). If the environment is subject to considerable change, and if the changes are unpredictable, it may be necessary to equip the agent with resources that it currently does not require to perform its task. Such a design would still be cheap in the sense used here. Depending on the adaptive requirements, even an Edelman-style system, based on selectionist mechanism would be considered cheap, much cheaper than preprogrammed systems, where all the potential situations would have to be foreseen. Having a system with general capabilities that can be exploited in specific situations, can be a cheap strategy.

And third, cheap means exploiting the constraints of the ecological niche. Above we have already seen that some behaviors can be achieved much more efficiently. An example is Horsewill's robot Polly, which exploits the fact that office floors are flat (see figure 3). Learning systems do not have to be universal: only very rarely is there a need in the real world to learn something odd like an XOR function. It has even been experimentally shown, that natural systems perform poorly on XOR learning tasks (e.g. Thorpe and Imbert, 1989). And it is hard to think of natural situations in which the ability to solve an XOR problem would confer an

advantage. "In general, if two cues both signal that food is about to arrive, when the two are present at the same time, the food is even more likely to appear!" (Thorpe and Imbert, 1989, p. 85). As a consequence much simpler neural networks may be used.

Focusing on cheap designs has the additional advantage that the limitations of a design, the ecological niches in which it will function, become immediately evident.

It is interesting to note that cheap designs in the sense discussed here imply ecological balance. Inflating one part, like building a huge brain while leaving sensors and effectors at the same level of complexity, will in any case be too expensive.

This view of cheap designs does not really have an analog in classical AI and cognitive science. There is no embodiment, there is no physics to be exploited, and there are no interesting interactions with the environment. The only overlap seems to be Occam's principle. But all the rest does not make sense in a classical perspective. Again, we see very clearly the fundamentally different view of intelligence endorsed here.

There is an interesting relation of the principle of "cheap design" to societies of animats. Often, tasks can be accomplished much cheaper by having a society of less sophisticated agents, rather than having one or only a few highly complex individuals (e.g. Mataric, 1995).

References supporting this principle include: Brooks, 1991; Horsewill, 1992; Franceschini et al., 1992; Pfeifer, 1993, 1995; Thorpe and Imbert, 1989.

4.5 Strategies, stances, metaphors

This category of design principles concerns the design process itself. Rather than constraining the designs of the agents directly as the set of principles outlined above, they provide suggestions on how to proceed. These principles are less well articulated and will not be discussed here. They include compliance with the design principles, taking the "frame-of-reference problem" into account (Clancey, 1991), incorporating constraints of the ecological niche, capitalizing on system-environment interaction, viewing the complete agent as a dynamical systems, etc.

5 Discussion

The design principles outlined above do not cover all the insights of the very rich field of animats. But we do believe that they capture a large part of the most essential aspects of what has emerged from pertinent research in the area. The principles described may seem somewhat vague and overly general, but they are enormously powerful as heuristics, providing guidelines as to what sorts of experiments to conduct next and what agents to design for future experiment. In order to achieve some degree of generality we have deliberately left out a lot of detail. These principles not only help us evaluate existing designs, but they get us to ask the right questions.

As mentioned initially we have not specifically discussed simulated evolution. Eventually, we may include some pertinent principles into our current set. In a number of places we have resorted to evolution for explanation. If evolutionary robotics gets to a stage where not only control architectures, but also morphology, sensors, effectors, and whole bodies can be evolved, it will be fascinating to see whether our principles also hold for these evolved creatures. A prerequisite is, of course, that the simulation environment reflects our laws of physics. Will these creatures also have value systems and self-organizing schemas? Will they also exploit the physics in interesting ways? If the creatures turned out to obey our design principles this would add additional force to them. But that remains to be seen.

In the future we might be looking for something more formal, than merely a set of verbally stated design principles. Eventually, this will certainly be necessary. As pointed out initially, the mathematical theory of dynamical systems is a promising candidate. But since we are dealing with "Fungus Eaters", i.e. complete multifaceted systems, it may be a while before we have a formal theory of "Fungus Eaters". This does by no means exclude productive use of formal methods to study specialized issues like learning algorithms, problems of mechatronics, etc.

What is needed right now is an in-depth discussion of these principles. They have to be revised and the list of principles has to be augmented. Moreover, an appropriate level of abstraction has to be found. It may turn out that the principles will be more useful if they are more concrete. However, that would imply the well-known trade-off between generality and direct applicability.

7 Conclusions

The design principles discussed in this paper communicate a view of intelligence and human cognition that is entirely different from the classical one endorsed by traditional AI and cognitive science. It seems that it would be premature to ask for a theory of autonomous agents. This is why we started with a set of principles that can help us formulate our beliefs about the nature of intelligence in a compact way.

Having a concise way of talking about our views of intelligence is extremely important since we want to convince researchers from other disciplines like psychology, biology, and neurobiology, that novel perspectives and directions can be expected from the animat field. The animate perspective may shed new light on old controversies. Examples are the conundrums involved in perception and categorization. While designing agents—as discussed above—is a fascinating and productive endeavor in itself, it is a highly creative tool for other scientific disciplines involved in the study of intelligence.

Let us conclude by saying that we hope to have made a small contribution towards Jean-Arcady Meyer and Anne Guillot's quest for theoretical advances and generalizations

Acknowledgments

This research was supported in part by grant # 20-40581.94 of the Swiss National Science Foundation. I would like to thank Christian Scheier, Dimitri Lambrinos, and Ralf Salomon for their inspiring discussions and valuable comments on the manuscript.

References

Ballard, D.H. (1991). Animate vision. *Artificial Intelligence*, **48**, 57-86.

Beer, R. (in press). The dynamics of adaptive behavior: a research program. To appear in: *Robotics and Autonomous Systems, Special Issue on "Practice and Future of Autonomous Agents"*, R. Pfeifer, and R. Brooks (eds.).

Braitenberg, V. (1984). *Vehicles: experiments in synthetic psychology.* Cambridge, Mass.: MIT Press.

Brooks, R.A. (1990). Elephants don't play chess. *Robotics and Autonomous Systems*, **6**, 3-15.

Brooks, R.A. (1991). Intelligence without representation. *Artificial Intelligence*, **47**, 139-160.

Brooks, R.A. (1994). Coherent behavior from many adaptive processes. In: D. Cliff, P. Husbands, J.-A. Meyer, and S.W. Wilson (eds.). *From animals to animats 3. Proc. SAB'94,* 22-29.

Brooks, R.A., and Stein, L.A. (1993). Building brains for bodies. Memo 1439, MIT Artificial Intelligence Laboratory, Cambridge, Mass.

Bushnell, E.M. and Boudreau, J.P. (1993). Motor development in the mind: The potential role of motor abilities as a determinant of aspects of perceptual development. *Child Development*, **64**, 1005-1021.

Clancey, W.J. (1991). The frame of reference problem in the design of intelligent machines. In K. van Lehn (ed.). *Architectures for intelligence.* Hillsdale, N.J.: Erlbaum.

Cruse, H. (1991). Coordination of leg movement in walking animals. In: J.-A. Meyer, and S.W. Wilson (eds.). *From animals to animat 1. Proc. SAB'90,* 105-119.

Dawkins, R. (1986). *The blind watchmaker.* London, UK: Penuin Books (1988) (first published by Longman).

Dennett, D. (in press). Cog as a thought experiment. *Robotics and Autonomous Systems, Special Issue on "Practice and Future of Autonomous Agents"*, R. Pfeifer, and R. Brooks (eds.).

Dewey, J. (1896). The reflex arc concept in psychology. *Psychol. Rev.*, **3** (1981) 357-370; Reprinted in: J.J. McDermott (ed.) *The Philosophy of John Dewey.* Chicago, IL: University of Chicago Press, 136-148.

Douglas, R.J., Martin, K.A.C., and Nelson, J.C. (1993). The neurobiology of primate vision. *Bailliere's Clinical Neurology*, **2**, No. 2, 191 - 225.

Edelman, G.E. (1987). *Neural Darwinism. The theory of neuronal group selection.* New York: Basic Books.

Franceschini, N., Pichon, J.M., and Blanes, C. (1992). From insect vision to robot vision. *Phil. Trans. R. Soc. Lond. B*, **337**, 283-294.

Gat, E. (in press). Towards principled experimental study of autonomous mobile robots. To appear in *Autonomous Robots.*

Harvey, I., Husbands, P., Cliff, D., Thompson, A., Jakobi, N. (1996). Evolutionary robotics: the Sussex approach. *Robotics and Autonomous Systems, Special Issue on "Practice and Future of Autonomous Agents"*, R. Pfeifer, and R. Brooks (eds.).

Hemelrijk, C.K., and Lambrinos, D. (1994). Peformance of two homing strategies in environments with differently distributed obstacles. *Perac-94*, 352-355.

Horsewill, I. (1992). A simple, cheap, and robust visual navigation system. In: J.-A. Meyer, H.L. Roitblat, and S.W. Wilson (eds.). *From animals to animats 2. Proc. SAB'92,* 129-137.

Kohonen, T. (1988). *Self-organization and associative memory.* Berlin: Springer.

Lambrinos, D. (1995). Navigating with an adaptive light compass. *Proc. ECAL-95,* 602-613.

Maes, P. (1991). A bottom-up mechanism for behavior selection in an artificial creature. In: J.-A. Meyer, and S.W. Wilson (eds.). *From animals to animats 1. Proc. SAB'90,* 238-246.

Maes, P. (1992). Behavior-based artificial intelligence. In: J.-A. Meyer, H.L. Roitblat, and S.W. Wilson (eds.). *From animals to animats 2. Proc. SAB'92*, 2-10.

Martinetz, T. (1994). Topology representing networks. *Neural Networks*, **7**, 505-522.

Mataric, M. (1995). Evaluation of learning performance of situated embodied agents. *ECAL-95*, 579-589.

McFarland, D. (1991). What it means for robot behavior to be adaptive. In: J.-A. Meyer, and S.W. Wilson (eds.). *From animals to animats 1. Proc. SAB'90*, 2-14.

McFarland, D., and Bösser, M. (1993). *Intelligent behavior in animals and robots.* MIT Press.

Meyer, J.-A., and Guillot, A. (1991). Simulation of adaptive behavior in animats: review and prospect. In: J.-A. Meyer, and S.W. Wilson (eds.). *From animals to animats 1. Proc. SAB'90*, 2-14.

Meyer, J.-A., and Guillot, A. (1994). From SAB90 to SAB94: four years of animat research. In: D. Cliff, P. Husbands, J.-A. Meyer, and S.W. Wilson (eds.). *From animals to animats 3. Proc. SAB'94*, 2-11.

Nisbett, and Wilson (1977). Telling more than we can know: verbal reports of mental processes. *Psychological Review*, **84**, 231-259.

Pfeifer, R. (1988). Artificial intelligence models of emotion. In V. Hamilton, G.E. Bower, and N. Frijda (eds.). *Cognitive perspectives on emotion and motivation* (Proc. NATO Advanced Research Workshop). Amsterdam: Kluwer, 287-320.

Pfeifer, R. (1993). Cheap designs: exploiting the dynamics of the system-environment interaction. Three case studies on navigation. In: Conference on *"Prerational Intelligence"*. Center for Interdisciplinary Research, University of Bielefeld, 81-91.

Pfeifer, R. (1994). The "Fungus Eater" approach to the study of emotion. *Cognitive Studies,* **1**, 42-57 (in Japanese). English version: The ``Fungus Eater'' Approach to Emotion: A View from Artificial Intelligence. Artificial Intelligence Laboratory, University of Zurich, Techreport #95.04.

Pfeifer, R. (1995). Cognition — perspectives from autonomous agents. *Robotics and Autonomous Systems*, **15**, 47-70.

Pfeifer, R., Schreter, Z., Fogelman-Soulie, F., and Steels L. (eds.) (1989). *Connectionism in perspective.* Amsterdam: Elsevier.

Pfeifer, R., and Scheier, C. (in press). Sensory-motor coordination: the metaphor and beyond. To appear in: *Robotics and Autonomous Systems, Special Issue on "Practice and Future of Autonomous Agents"*, R. Pfeifer, and R. Brooks).

Pfeifer, R., and Verschure, P.F.M.J. (1992). Distributed adaptive control: a paradigm for designing autonomous agents, in F.J. Varela, and P. Bourgine (eds.) *Proc. ECAL-92*, 21-30.

Pfeifer, R., and Verschure, P.F.M.J. (1995). The challenge of autonomous systems: pitfalls and how to avoid them. In L. Steels and R. Brooks (eds.). *The Artificial Life Route to Artificial Intelligence.* Hillsdale, N.J.: Erlbaum, 237-263.

Scheier, C., and Lambrinos, D. (1996). Categorization in a real-world agent using haptic exploration and active vision. *Proc. SAB'96.*

Scheier, C., and Pfeifer, R. (1995). Classification as sensory-motor coordination: a case study on autonomous agents. *Proc. ECAL-95,* 657-667.

Smith, L.B., and Thelen, E. (eds.) (1993). *A dynamic systems approach to development. Applications.* Cambridge, Mass.: MIT Press, Bradford Books.

Smithers, T. (1995). On quantitative performance measure of robot behaviour. *Robotics and Autonomous Systems*, **15**, 7-133.

Steels, L. (1992). The PDL reference manual. VUB AI Lab memo 92-5.

Steinhage, A., and Schöner, G. (in press). Self-calibration based on invariant view recognition: Dynamic approach to navigation. To appear in: *Robotics and Autonomous Systems, Special Issue on "Practice and Future of Autonomous Agents"*, R. Pfeifer, and R. Brooks (eds.).

Thelen, E. and Smith, L. (1994). *A dynamic systems approach to the development of cognition and action.* Cambridge, Mass.: MIT Press, Bradford Books.

Thorpe, S.J., and Imbert, M. (1989). Biological constraints on connectionist modelling. In R. Pfeifer, Z. Schreter, F. Fogelman-Soulié, and L. Steels (eds.). *Connectionism in Perspective.* Amsterdam: North-Holland, 63-92.

Toda, M. (1982). *Man, robot, and society.* The Hague, Nijhoff.

Webb, B. (1994). Robotic experiments in cricket phonotaxis. In. D. Cliff, P. Husbands, J.-A. Meyer, and S.W. Wilson (eds.). *From animals to animats 3. Proc. SAB'94,* 45-54.

Wilson, S.W. (1991). The animat path to AI. In: J.-A. Meyer, and S.W. Wilson (eds.). *From animals to animats 1. Proc. SAB'90,* 15-28.

Human Simulation of Adaptive Behavior: Interactive studies of pursuit, evasion, courtship, fighting, and play

Philip W. Blythe, Geoffrey F. Miller, and Peter M. Todd
Center for Adaptive Behavior and Cognition
Max Planck Institute for Psychological Research
Leopoldstrasse 24, 80802 Munich, Germany
Email: blythe@, miller@, ptodd@mpipf-muenchen.mpg.de

Abstract

To understand more about how animate motion is generated and perceived, we need quantitative analyses of motion trajectories from organisms interacting in various important adaptive tasks. Such data is difficult to obtain for most animals, but one species provides a ready source. We have developed software that allows human subjects to generate such motion data by interacting across a computer network in on-screen pursuit and evasion, fighting, courtship, and play. Each subject uses a mouse to control a "bug" that moves in a 2-D environment with another bug controlled by a second remote subject. We have visualized and analyzed the resulting motion data for each task in several ways: 3-D space-time plots of the trajectories themselves, scatterplots of one bug's positions relative to the other, and statistical measures of trajectory parameters including velocity, vorticity, and energy. All of these methods partly distinguish between the different motion categories. We have also gathered naive subjects' categorizations of these trajectories to explore further the information that makes them more or less distinguishable. Having human subjects perform these kinds of scenarios can lead to better techniques for analyzing, comparing, and designing the motion capacities of simulated agents.

1 Introduction

Behavior is motion. Consequently, simulation of adaptive behavior (SAB) research typically implies the simulation of motion. But research in this field too often relies on the creator's subjective impressions of whether a certain behavioral trajectory generated by a simulated agent looks sufficiently "lifelike" or "animate" in a particular task. We suggest a simple, novel method for overcoming this problem: have people simulate the desired motion trajectories themselves, and use this data as the basis for judging a system's performance.

This method requires human subjects interacting over a computer network to perform the same behavioral tasks that SAB researchers impose on their simulated animals. Such experiments can yield rich motion-trajectory data, as well as subject comments and reactions that can be used in several ways. First, we can check that the imposed tasks make sense to a 100-billion-neuron human; if not (and assuming that humans should be good at such tasks), they may not make sense to a much more cognitively-challenged 10-neuron simulated bug. After this simple reality-check, we can compare the human-generated motion trajectories to those generated by our simulated agents, using a whole arsenal of quantitative measures, to assess how appropriate the simulation's behavior is. We can also use the human-generated data as training exemplars for simulated evolution or learning to shape our agents' behavior in the first place. Finally, we can study human responses to motion generated by other humans and by simulated agents, to investigate the cues our species uses to perceive and interpret animate motion – and, by extension, the cues that simulated agents should both generate and perceive themselves.

We are particularly interested in this last use of the tool we have developed: understanding how humans and other animals categorize the different functional types of animate motions we observe in the world. We all can tell the intentions of predators or prey, courters or desired mates, by the patterns of positions in space that they occupy over time. What cues do we use to make these judgments? How can we tell when one organism is fleeing, stalking, or playing with us or with another organism? To find out, we need a set of motion patterns that we can have people categorize, and then analyze to find their distinguishing characteristics. But such data is somewhat hard to come by. The literature in both biology and psychology is full of studies of both long-range animal navigation, migration, and commuting, on the one hand, and small-scale limb movements, on the other. However, rather little recorded data on behavioral trajectories exists at the scales of interest in between these two extremes. So instead we decided to generate our own, using the animals at hand: the other people in our own lab.

But rather than attach beacons to the tops of their heads and film them from above as they go about their

daily activities, we put pairs of people into a simple, highly constrained computer world setting in which they can only interact through movement. Specifically, each person controls their own two-dimensional "bug" in an on-screen environment in which both bugs appear. These human subjects are instructed to engage in various tasks via their bugs – in other words, to "simulate" through their behaviors the very motion types we are interested in. We collect the data our subjects generate in this way, and analyze it to find the cues that distinguish one category of motion from another. We also let other subjects "analyze" this data as well, by showing them portions of previously-generated trajectories and seeing if they can tell what motion category each belongs to. By studying what trajectory snippets are more or less distinguishable from each other in this way, we can get a better focus on what trajectory features are important cues for the various motion categories.

Part of the application of this work will be in the design of simulated or robotic autonomous agents that move in natural ways that we can identify and attribute specific goals and functions to. One way to test whether our newly-minted motion algorithms truly capture the relevant aspects of animate behavior is to see if people find them convincing. To do this, we have devised a kind of behavioral Turing test, in which one human subject controls a bug interacting with a second bug as usual. But in this case, the subject must decide if the other bug is also controlled by a person, or by an artificial algorithm, based solely on their intertwined motions. If the human subjects cannot tell when they are interacting with a program or another person, then our algorithms will be deemed successful, by being able to produce the appropriate cues of animate motion for this particular task.

This paper reports the results of our human-simulated-behavior approach. Section 2 reviews the biological and psychological foundations of animate motion research. Section 3 describes our methods, including the computer task environment and our experimental procedure. Section 4 presents our results, including motion trajectories from six different tasks, various measures for distinguishing each trajectory type, and subject categorizations of previous trajectories. Finally, section 5 describes possible applications of this method, both for SAB research directly, and for investigating the psychology of animate motion perception. We offer this approach as a useful adjunct to existing SAB research methods – a way of keeping ourselves from being deluded by our human tendency to attribute animacy and intelligence to almost anything that moves, and a way of understanding how we can effectively manipulate that tendency to make our animats look more animate.

2 Theoretical Foundations

There has been a great deal of work on animate motion from a variety of theoretical angles. Perceptual psychologists following Johansson [Johansson, 1975] have run hundreds of studies showing that humans are rather good at perceiving and classifying animate motion from point-light displays – videos that show only the traces of tiny lights attached to various parts of a moving human or animal [Cutting and Kozlowski, 1977, Dittrich, 1993, Mather and West, 1993]. Neurophysiologists have identified various temporal lobe areas specialized for perceiving these animate motion displays [Oram and Perrett, 1994]. Developmental psychologists have stressed the high degree of innate preparedness that infants show for recognizing animate motion [Gelman, 1990, Premack, 1990]. Neuroethologists have studied the neural circuits for attack and escape behaviors of invertebrates in great detail, sometimes recording motion trajectories but seldom doing quantitative analysis on them [May, 1991]. Behavioral biologists have studied animal navigation and foraging trajectories, spanning large distances and extended time periods. Some psychologists have attempted to apply dynamical systems theory to model the dynamics of high-level cognition [Port and van Gelder, 1995], but only a few researchers have applied this theory to the lower-level dynamics of animate motion itself [Beer, 1995, Cliff et al., 1993].

We have found little work, however, to guide us in analyzing the perceptual cues that distinguish one functional category of motion from another. The first step in finding these cues is to establish the basic set of motion categories that concern animate species, with which the cues will be associated. Animate motion is used for a rather small number of functions that can be largely deduced from the implications of natural selection and sexual selection. Basically, animals evolve to interact adaptively with various "fitness affordances" in their environments – things that are likely to affect the replication of their genes [Miller and Freyd, 1993, Todd and Wilson, 1993]. Positive fitness affordances, like food and sexual mates, promote survival or reproduction; negative fitness affordances, like predators, pathogens, parasites, and sexual competitors, interfere with survival or reproduction. Animals evolve sensory-motor systems to approach the positives and avoid the negatives. If two animals offer mutually positive yields, mutual approach results; if they threaten mutually negative yields, then mutual avoidance results. Finally, if instead, two animals have a conflict of interest, more complex interactions can result, transforming simple approach into persistent pursuit, and simple avoidance into desperate evasion [Miller and Cliff, 1994].

From the above arguments, it follows that the fundamental categories of animate interaction are mutual approach (boring), mutual avoidance (also boring), and pursuit and evasion (interesting and unpredictable). In the survival domain, pursuit and evasion usually occur between predators and prey, or between fighting dominant and submissive conspecifics; in the reproductive domain, pursuit and evasion usually occur between male and female in the roles of courter and courtee (or sexual harasser and harassee). Thus, any particular animal will need to master some subset of five basic categories of

animate motion: pursuing, evading, fighting, courting, and being courted. To these categories we also add a sixth, play, which is widely used as a way of learning the skills to master the other five movement types. Between species, these motion categories can vary dramatically in the degree of cognitive complexity that they require, but they remain functionally similar over the entire animal kingdom. We now briefly consider the characteristics of each of these classes in turn.

Pursuit: Animals move towards objects they desire. If the desired object is inanimate, we have a degenerate case of goal-directed behavior. But if the object is animate and does not want to be exploited as a fitness affordance (e.g. as food or as a mate), then it will move away (evade). Pursuing often benefits from a predictive strategy, as opposed to reactive approach: more successful pursuers try to anticipate where their opponent will go, based on that opponent's current heading and intentions and the environmental constraints they face (obstacles and boundaries), and then try to cut them off [Reynolds, 1994].

Evasion: Animals move away from things that threaten them. Again, if the threatening object is inanimate, we have a degenerate case of obstacle avoidance, or one-step "evasion". If the threat is animate, however, and does not wish to be evaded, then it will pursue, and sustained evasion becomes necessary. Evasion often favors strategies of deceptive feints and lunges, and/or unpredictable, "protean" zig-zagging [Driver and Humphries, 1988]. Also, evaders must avoid being boxed in by environmental features that limit their trajectory options.

Fighting: Animals of the same species often fight over fitness affordances such as territories, resources, sexual mates, and social status. Fights are tricky because both animals must combine pursuit and evasion, attack and defense, in a way that intimidates or overcomes the opponent, without risking injury or death to themselves. Because animal bodies are heterogeneous, with some parts specialized for attack and other parts vulnerable to injury, fighting includes a great deal of precise body-positioning.

Courting: Animals – usually males – move towards members of the opposite sex – usually females – that they want to mate with. But because selective mate choice is almost always imposed by the opposite sex, simple approach is almost never enough. Instead, mate-seeking animals often evolve extremely complex courtship behaviors with special features designed to display their health, strength, size, status, intelligence, or creativity [Andersson, 1994, Miller, 1996]. These displays are usually produced close enough for the desired mate to perceive them, but not too close, lest they flee. After some display time, ranging from seconds (for some insects) to years (for some humans), if the desired mate signals their interest somehow, the final approach and copulation can occur.

Being courted: Animals sought after as mates – usually females – have strong incentives to select among their suitors quite carefully, because the genetic quality of the suitors they choose to mate with will determine half the genetic quality of their offspring. Random mating is stupid mating. The task when being courted, then, is to express enough interest to elicit informative courtship behavior from various suitors, but not to express so much that they skip courtship altogether and try to move straight to copulation. Thus, being courted requires a delicate balance between interactive encouragement and coy reticence. Courted animals usually maintain enough proximity to their suitors that they can see what's going on, but do not get close enough to risk real sexual harassment or rape.

Playing: Play is basically a catch-all category in which young animals might practice all of the above movement types, using various play signs to indicate that they are pursuing, evading, courting, or fighting without real lethal or sexual intent [Fagen, 1981]. In basic play, animals repeatedly switch roles between pursuer and evader, or attacker and defender. In more complex play characteristic of large-brained primates, animals may interact in more abstract ways with imaginary partners or mutual mimicry.

3 Methods for Motion

3.1 Motivation for our design

Real animate motion in natural environments filled with other agents and obstacles is too complex to analyze very clearly. Our experimental methods were designed to collapse this complexity down to manageable but still informative dimensions – specifically, two spatial dimensions plus time. We decided to squeeze the camel of animate motion through the needle's eye of a two-person computer-based interaction occurring in a featureless, two-dimensional on-screen environment. Based on our theoretical analysis in the previous section, we picked just six tasks for subjects to do in this environment, solely by generating movement patterns through the horizontal and vertical positions of a computer mouse. Although these simplifications might seem extreme, we found that subjects had no trouble identifying with the expressionless one-inch bug they controlled on-screen, becoming highly motivated to guide its motion as best they could to cope with the varied challenges posed by the other subject-controlled bug.

3.2 Experimental setting

Twenty subjects participated in 10 trials of our interactive games. During each experiment, two subjects (hereafter called A and B) were run at the same time in different rooms. Each subject was seated at a u*ix workstation, and told that they would be engaging in a series of interactions between their own bug and another bug that appeared on the screen.

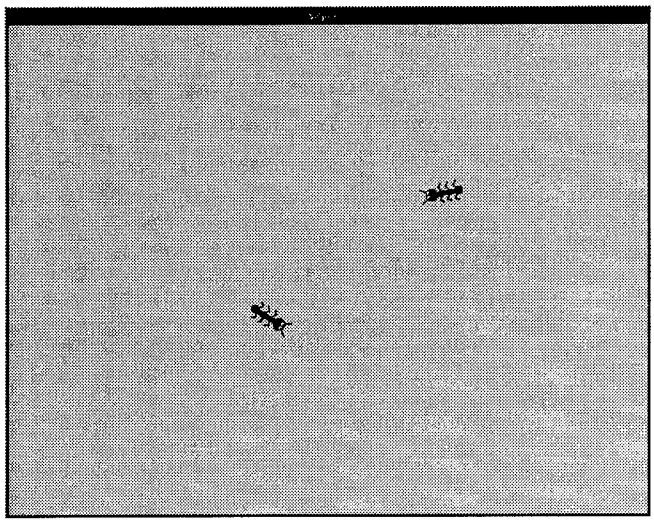


Figure 1: Example bug environment

Throughout the trials, the chief experimenter seated at a third workstation started and stopped each trial and watched a display of both bugs interacting. During an initial two minute practice period, subjects learned how to control their bug using their mouse, getting used to the physics we instantiated (see section 3.4). Then, subjects (who were anonymous to each other) completed six experimental trials of 120 seconds each (180 seconds for courtship trials). The trials were ordered as follows:

(1) A pursues B, B evades A: Subject A was instructed to try to pursue and hit the other bug as quickly and as often as possible, and was told that the other bug would try to avoid being hit. Subject B was instructed to try to avoid being hit, and was told that the other bug would try to pursue it.
(2) B pursues A, A evades B: As in (1), with roles reversed.
(3) A courts B, B is courted by A: Subject A was instructed to court the other bug, by interacting with it in any way that it might find interesting, exciting, or enticing. Subject B was instructed to play the role of being courted, and to show interest or disinterest, or to elicit further displays in any way desired, in response to what the other bug is doing.
(4) B courts A, A is courted by B: As in (3), with roles reversed.
(5) Fighting (same for A and B): Subjects A and B were instructed to attack the other bug from behind, while at the same time avoiding being attacked in return.
(6) Playing (same for A and B): Subjects A and B were instructed to play with the other bug in whatever manner they wanted.

3.3 The Appearance of the Environment

For each trial, a new environment window fills each subject's 21 inch computer screen. The window is a featureless light-tan rectangular space, representing a top-down view of a simple environment. In that space are two bugs, identical except for color (one red and one green). From the subjects' top-view perspective, each bug is about an inch long on-screen (see figure 1). This top-down view – rather than a "bug's-eye view" – is justified for our current studies by the fact that humans can (and do) judge the motion trajectories of other animals from an elevated down-looking viewpoint. In future studies, we will consider the problem of judging motion categories from self-relative motion (i.e. from a bug's-eye perspective), but this will require a different method for generating trajectories.

The environments used in all trials had constraining walls around the perimeter, which strongly affects the resulting motion trajectories. The bugs also had a different preprogrammed reflexive behavior for each task, which fired whenever contact was made: for pursuit/evasion and fighting, a low-frequency temporary grappling ensued; for courtship, a higher-frequency interaction was used; and for play, a simple bouncing repulsion occurred. All of these simulations of localized behavior were developed to encourage a more realistic impression of the assigned tasks, as well as forming a type of natural payoff for good performance. In general, subjects found the tasks highly engaging, despite the cartoonish appearance of the bugs, the emptiness and flatness of the environment, and the indirect way that their mouses controlled the bugs via the target positioning. The interactiveness of animate motion alone was enough to give the experiment a sense of realism.

3.4 Bug Physics

Subjects move their bugs using their mouse. Rather than direct "screen cursor" control that instantly tracks mouse movement, the mouse moves a "target cursor," invisible to subjects, that specifies where the bug should head at each moment. The greater the distance between bug and target, the faster the bug moves towards the target, according to the following second order discretized equation, integrated by Euler approximation[1]:

$$\{\dot{x}\}_t = C(\{x_p\}_t - \{x\}_t) + G\{\dot{x}\}_{t-1} \qquad (1)$$

where $\{x_p\}$ is the position of the mouse-controlled target and $\{x\}$ denotes the bug's position vector at timestep t, and C and G are constants.[2] The bug also has limited linear and rotational speeds. In practice, subjects moved their mouses around quite quickly in pursuit, evasion, and fighting, usually sacrificing fine positional control for raw speed. In courtship and play, subjects moved their bugs more slowly and deliberately.

Figure 2 illustrates the relevant parameters in the simulation, which fall into two major groups. The first are the dynamic parameters used for simulating the motion of the bugs:

[1] A high refresh time of 100-200Hz is required in the simulation dynamics, even though display rates need not be as high, to prevent accumulation of round-off errors in the integrations and ensuing spurious motion effects.

[2] The values used were $C < 0.02$ and $G < 0.6$, which gave the best simulated effect of second order motion without the need for a regulator law.

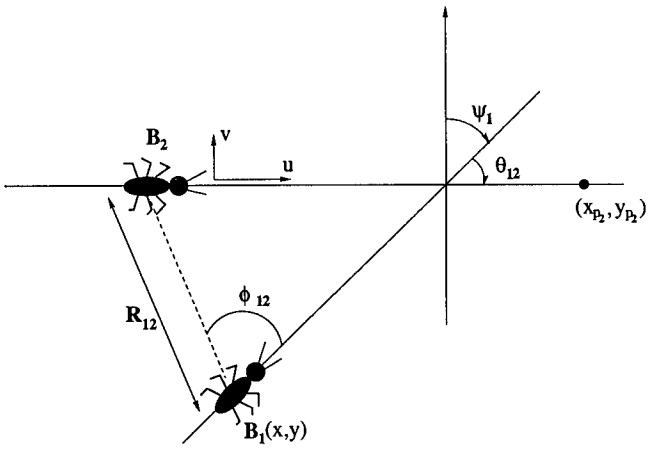


Figure 2: Parameters of the bug environment

$\{x, y\}$: bug position in the environment
$\{x_p, y_p\}$: mouse-controlled target position
ψ: bug heading relative to the environment
$\{u, v\}$: bug velocities relative to the bug's heading

The second group of parameters capture the relationships between the two bugs that can strongly influence the subjects' controlling behavior. As such, these parameters are probably more important for distinguishing and producing different types of animate motion than the first group (a hypothesis we begin to test in section 4.4):

R_{ij}: current distance between the two bugs (i.e. their centers)
ϕ_{ij}: the perceptual angle between one bug's current heading and the other bug's location
θ_{ij}: the relative angle between the targeted headings of the two bugs

4 Results

4.1 Space-time plots

To visualize the motion trajectories generated in the interactive tasks, 3-D plots of position over time were constructed for each subject pair and each task (see figures 3-6). In each plot, the horizontal coordinates directly correspond to the position of each bug on-screen, and the vertical coordinate is time, running from the start of the trial (bottom) to a time 90 seconds later (top). Thus, the flatter the slope of a plotted line, the faster a bug's position is changing over time – that is, the higher its velocity. Relative distance and relative heading between the bugs are also implicit in this representation. Each plot contains the two trajectories (one for each subject) simultaneously generated in a single trial. Because pursuit and evasion are reciprocal tasks, as are courtship and being courted, we have only shown one plot for each, capturing both roles.

By comparing plots for sample runs from the different tasks, several distinctive features are immediately apparent. In pursuit and evasion (figure 3), one can see

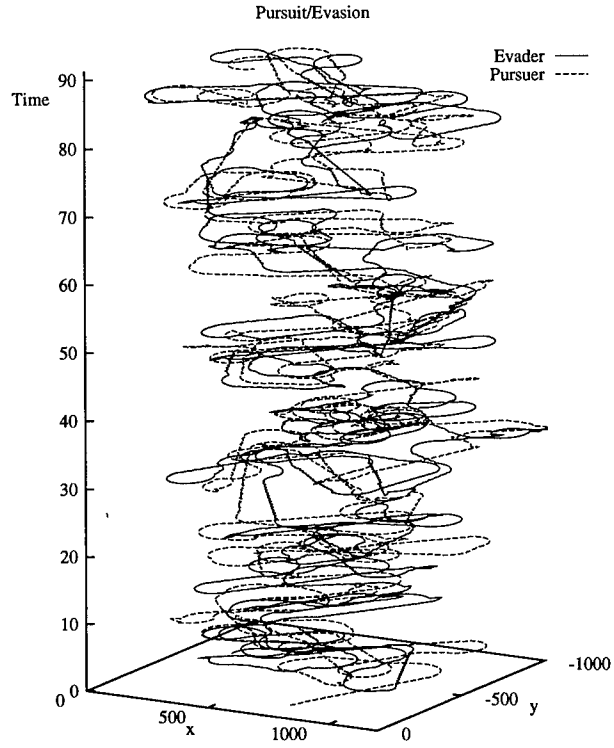


Figure 3: Sample chasing trajectories

very flat (very high speed) movements extending over a greater area of the environment than during courtship (figure 4) or play (figure 6). Pursuit/evasion (figure 3) and fighting (figure 5) show similarly high speeds and large amounts of turning and looping, but in fighting the average distance between bugs is smaller, and the looping is tightly intertwined. In courtship (figure 4), the courter moves much more than the often stationary courtee, sometimes circling, and occasionally engaging the courtee in little bursts of pursuit and evasion. Only a few body contacts (where the trajectories meet) are apparent in courtship. Play (figure 6) looks like a combination of pursuit, evasion, fighting, and courtship; in the trial shown, one of the subjects was much more active than the other.

4.2 Scatterplots of relative position

Of all the on-screen information that is perceptually available to subjects, only a few parameters might actually be used to control the bugs. We hypothesized that subjects use the position of the other bug relative to their own bug's current position and heading as input to simple motion-control heuristics. We also expected that, even if relative position is not the only information used in bug control, scatterplots of this information should show up differences between the motions generated in different tasks.

To represent this information, polar-coordinate scatterplots were constructed for each task, including all data

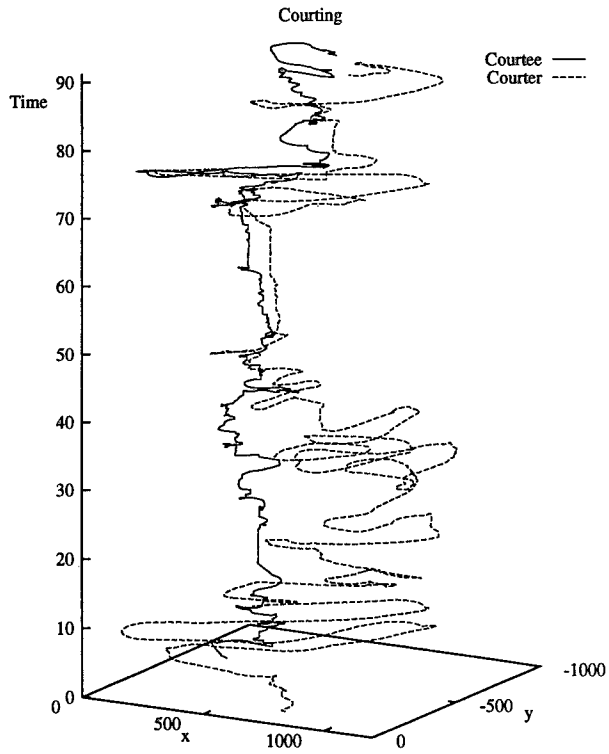


Figure 4: Sample courting trajectories

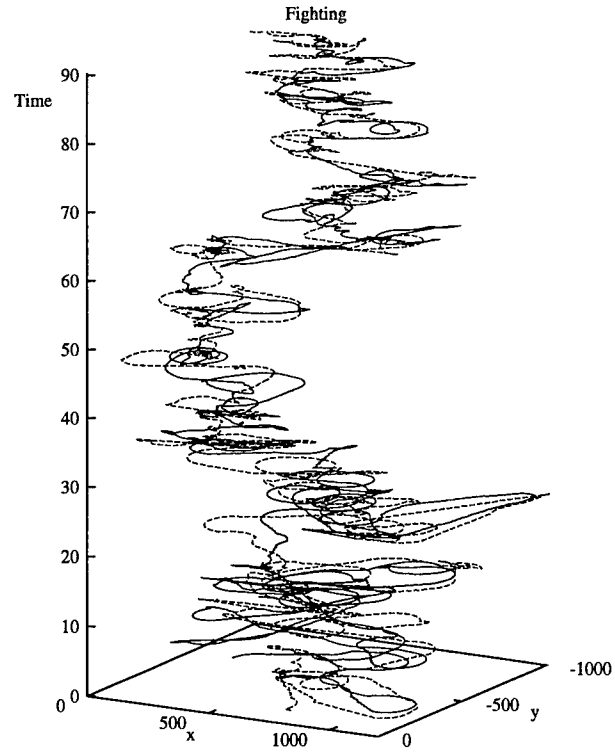


Figure 5: Sample fighting trajectories

from a number of randomly-selected subject pairs. For each subject's bug at each point in time, the relative position of the other bug is represented as a single dot at a certain relative angle and distance. One's own bug is positioned at the origin, facing rightward. If the other bug were close and straight ahead, for example, it would be plotted as a point just to the right of the origin on the horizontal axis. Thus, the scatterplots portray the statistical distribution of where the other bug stands in relation to one's own bug, as a result of the strategic interactions between subjects in each task.

When pursuing (figure 7), subjects were usually able to keep the other bug in front of their own bug, and not very far away. However, the pursuers rarely achieved a "kill": the roughly heart-shaped (cardioid) distribution of relative positions grows sparse around the origin, where actual contact points would appear. The distinctive shape of this distribution is mirrored in evasion (figure 8), where subjects were able to keep their pursuers behind them. The results for fighting (not shown) are strikingly different: although both subjects were trying to pursue the other bug's tail with their own head, we do not see the same cardioid distribution as for pursuit. Rather, the scatterplot is almost indistinguishable from a 2-D normal distribution, with complete radial symmetry and the highest density of points right around the origin, corresponding to close-body contact. Finally, courtship and play both included many distinctive long looping structures, unlike anything apparent in pursuit, evasion, or fighting.

4.3 Quantitative measures of trajectory parameters

In addition to the rather qualitatively interpreted 3-D space-time plots and the relative position scatterplots, we have also applied a set of quantitative measures to the trajectory data. Our hope is that one or a combination of these measures can be used to distinguish motion generated in the different tasks. The measures applied so far are:

- Average Velocity : computed for each bug over an entire trial
- Kinetic Energy : the integrated expended kinetic energy of a bug over an entire trial
- Vorticity : the summed total of the absolute changes in heading of a bug over an entire trial, indicating the amount of twisting and turning it did
- Radial Distance : distance between the two bugs at each instant, time-averaged over an entire trial
- Relative Heading : calculated as $cos(\theta)$ at each time step, averaged over entire trial; +1 indicates both bugs are always facing the same direction, and conversely -1 indicates they always face each other
- Relative Angle : calculated as $cos(\phi)$, as above; +1 indicates the other bug is always in front, and -1 means always behind

Table 1 compares these measures, averaged across all subjects (with standard deviations) for each different task. Velocities were very high in pursuit, evasion,

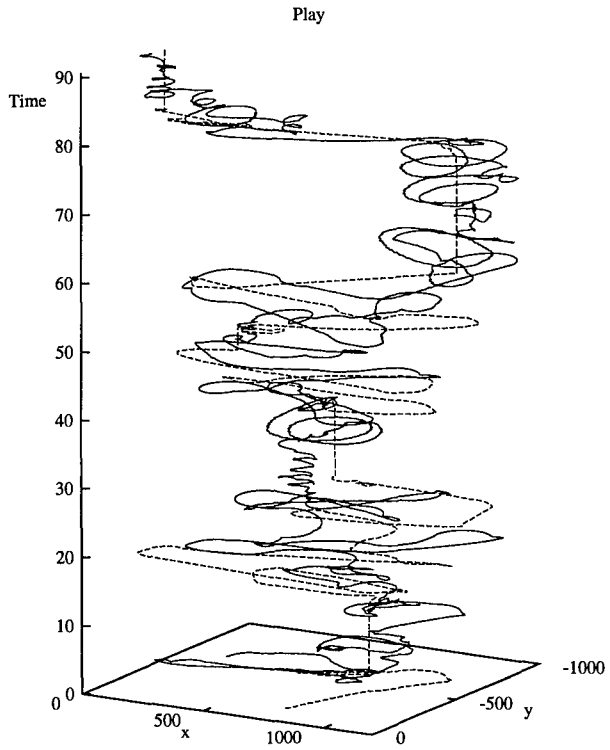


Figure 6: Sample playing trajectories

and fighting, whilst significantly lower in courting, and lowest when being courted. Also, the low variance of velocity in pursuit and evasion highlights this as a more valid cue over all subjects for these tasks. Expended kinetic energy creates similar distinctions to those made by the average velocity, although the deviations are somewhat higher in the courtship tasks, suggesting this as a slightly less reliable indicator. Also, the vorticity patterns are distinctive, with fighting involving the largest amount of turning and looping, followed closely by courting, with pursuit, evasion, play, and being courted all showing significantly less turning.

The next three measures capture the relational cues between the two bugs. The radial distance measurement fails to make any strong distinctions, other than the consistent close proximity of fighting behavior. However, the relative heading and angle parameters are more informative. Pursuit and evasion are characterized by the two bugs facing the same direction, with the evader predominantly in front. During fighting the two bugs tend to point in opposite directions, largely while attempting to loop around behind each other. Finally, the courting bugs spend a lot of time looking at their desired mates, but this is not so well reciprocated – a depiction of real life?

The failure of these parameters to make strong distinctions between some of the tasks is in sharp contrast to the results of the recognition experiments (see section 4.4), which suggests that time averaging of

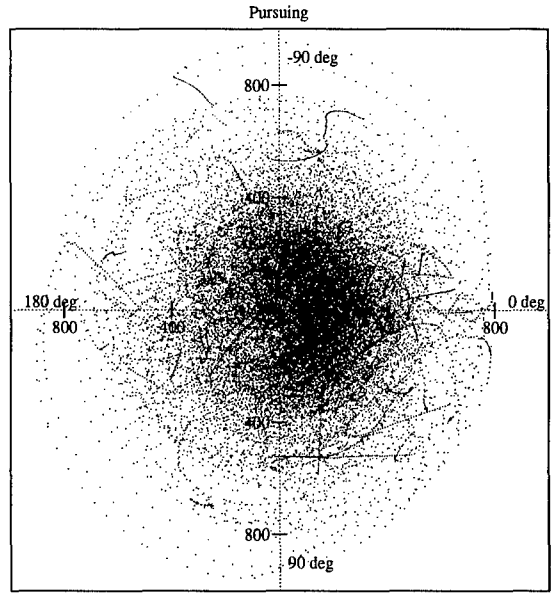


Figure 7: Relative heading / distance plot for chasing

these parameters may not be capturing the essential information content. We are currently developing further measures based on more sophisticated dynamical analyses in the frequency domain, which will hopefully identify the temporal content that people and animals use to categorize animate motion types.

	Pursuing	Evading	Courting	Courted	Fighting	Playing
Velocity	752 ±45	751 ±52	326 ±154	228 ±134	684 ±99	506 ±97
Kinetic Energy	591 ±49	592 ±55	195 ±137	132 ±109	517 ±105	346 ±92
Vorticity	8.05±1.02	7.82±1.82	9.87±4.11	7.31±4.16	10.3±1.60	7.28±1.56
Radial Distance	326 ±69	326 ±69	278 ±61	278 ±61	247 ±28	306 ±48
Rel Heading	0.16±0.14	0.16±0.14	-0.02 ±0.13	-0.02 ±0.13	-0.19 ±0.08	-0.04 ±0.08
Rel Angle	0.33±0.11	-0.47 ±0.09	0.16±0.19	0.04±0.28	-0.07 ±0.11	0.02±0.10

Table 1: Quantitative parameters of trajectories

4.4 Subject categorizations of generated trajectories

To test whether or not the subject generated trajectories are good examples of different motion categories, we ran a further experiment to test the ability of naive subjects to correctly identify and categorize the trajectories back into the original six behavioral classes. Ten subjects were given thirty consecutive movie snippets of the previously generated motion patterns, shown in a randomized order. They were then instructed to select one out of the six given motion categories which best matched that behavior. Table 2 shows the resulting confusion matrix, where the leading diagonal represents correct categorizations (note that the rows denote the actual categories, and the columns denote

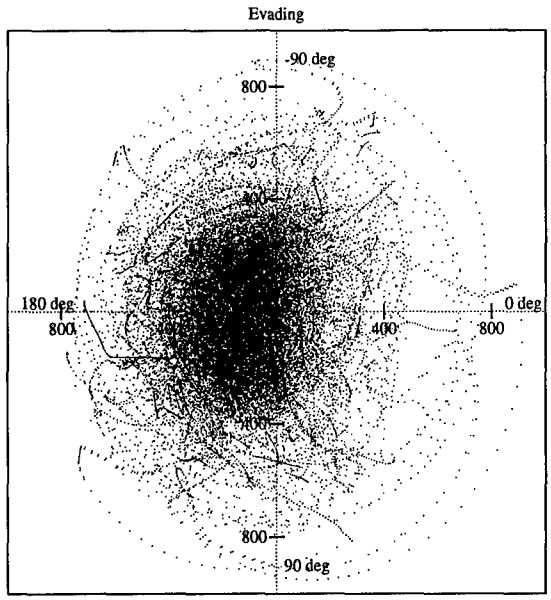


Figure 8: Relative heading / distance plot for evading

the subject selections). From a cursory glance, one can see confusion between chasing and play, play and fighting, as well as play and courtship. This fits well with the theoretical arguments of play behavior given in Section 2. Even though only 49% of responses were correct, this is significantly above chance level (16.7%).

We have also run a variation of this categorization experiment, where only one of the two bugs was visible in each trial. The aim of this experiment was to find out how important the relational cues are in determining the behavioral judgments. Table 3 gives the corresponding confusion matrix from this study (again for 10 subjects making 30 judgments), showing the increased confusability of pursuit and evasion, courting and being courted, and the continued confusion of play. Here, the percentage of correct responses dropped down to 30%, which indicates the relative importance of the relational cues in making behavioral distinctions.

Actual \ Choice	Pursuing	Evading	Courting	Courted	Fighting	Playing
Pursuing	29	1	1	0	9	17
Evading	5	20	0	5	6	7
Courting	2	0	38	2	4	8
Courted	0	2	5	29	4	6
Fighting	5	6	3	3	10	23
Playing	2	0	9	10	7	21

Table 2: Categorization confusion matrix with two bugs visible

Actual \ Choice	Pursuing	Evading	Courting	Courted	Fighting	Playing
Pursuing	15	8	12	3	7	7
Evading	13	14	9	1	5	6
Courting	2	5	16	17	5	9
Courted	5	7	4	21	2	5
Fighting	7	4	13	0	13	14
Playing	5	4	10	5	14	10

Table 3: Categorization confusion matrix with only one bug visible

4.5 Pilot studies of a motion Turing test

As a direct contrast to the human "bugjects," we have developed a series of simple robot bugs to perform certain interactive behavioral tasks, such as pursuit, evasion and courtship. A primary motivation for this is to generate "un-animate" behavioral data that can be used as a comparative baseline for determining exactly what constitutes convincing animate motion. Typically, the driving algorithms for these robot bugs use the position and relative distance of the other bug to determine its new trajectory. Though we have only created extremely simple algorithms at this point (including "narky," who evades by heading toward the opposite quadrant of a pursuing bug, and "peter," who courts by heading to positions at small random offsets from the courted bug), they are nevertheless convincingly animate when run against a real human-controlled bug.

To test just how convincing some of these simple strategies might be, we have devised a purely motion-based Turing test, in which a robot bug replaces one of the human subjects in a trial. After the trial, we ask the human subject whether they thought they were interacting with another human-controlled bug, or a computer-controlled robot bug. In one instance of this test, the robot bug algorithms were judged "human" by a subject in both the pursuit and evasion tasks. In another case, a subject spontaneously concluded that she had run all six trials – including courtship – against an automaton, when in fact they were all against a slightly embarrassed human subject. Of course, these two cases are merely illustrative, but even at this stage we have seen that this kind of a motion-based Turing test can be a powerful tool for gathering human judgments.

We have also created a further Turing test-like scenario for investigating the cues that are crucial to animate motion. For this scenario, we place subjects in front of a display on which they see the interactions of two bugs, played in real time. Subjects are purely passive observers in this case, watching the actions of others without controlling a bug of their own. We can show these subjects runs that are generated by two humans, by one human and one computer algorithm, or by two algorithms interacting with each other. We then ask the subjects to tell us what kind of entity was controlling each bug. In such runs, it quickly becomes clear that two

simple algorithms playing against each other look very mechanical and stereotyped, but a robot bug interacting with a human-controlled bug can look very animate. Essentially, by basing its trajectory on the motions of a human-controlled bug, the robot bug superimposes the animacy of the human onto its own movement patterns. This effect points up the need for very careful control in studying the generation of interactive animate motion, to know when we are seeing the result of the algorithm, and when the influence of the environment (including other agents).

5 Further applications

The software that we have developed to elicit human simulation of adaptive behavior can be put to many uses. This system should be available by the time of publication for use and development by other SAB researchers in a variety of domains – contact the first author for further information. Here, we outline four possible applications.

5.1 Qualitative human pre-testing of simulated environments and tasks

Researchers can run themselves as subjects to better understand the tasks they are actually setting for their simulated animals. In our experience, much of the (unpublished) wasted effort and false starts in doing SAB research (especially simulated evolution of animal nervous systems) come from misunderstanding how one's simulated creatures actually experience their environment and their intended tasks. (Counterexamples showing the usefulness of "test-driving" an animat environment by viewing it from the animat's perspective include [Blumberg, 1994, Cliff et al., 1993, Terzopoulos et al., 1994].) Simulated evolution has a maddening tendency to find all of the trivial, degenerate "cheats" that allow animats to fulfill the letter but not the spirit of some adaptive task. We suggest that researchers might often save weeks of computer time by running themselves first, as "pilot animats," to see what strategies *they* learn to solve the relevant task. Since many animat tasks are becoming more interactive, either based on predator-prey interactions, communication during courtship, or cooperative social behavior, network-based human experiments can be particularly useful. One can then better understand how the environment is perceived by the animat, and how it differs from the environment as imagined by the researcher.

5.2 Quantitative comparison of human- and animat-generated trajectories

SAB research has always paid at least lip service to the necessity of comparing simulated behavior to real animal behavior. But it is usually prohibitively difficult to get real animals to run a perfect analog of one's simulated task. By contrast, it is easy to get human subjects to run perfect analogs, by having them play a computer-game version of the behavioral task, often using much of the same simulation code. If the resulting motion trajectories (or other behaviors) generated by human subjects are represented in the same data format as the motion trajectories generated by animats, comparisons become easy, using a variety of quantitative measures. We have outlined a few such methods in this paper. Many more are possible. If the human and animat trajectories are indistinguishable according to such measures, they may also be indistinguishable qualitatively to human judges. That is, the animats could be considered to have passed a weak version of the Turing Test, using motion trajectories rather than the teletext as the arena of comparison.

5.3 Human-generated motion trajectories as training exemplars

Many simulated evolution and learning systems must provide animats with some form of feedback about their performance, either in the form of a fitness value used by a genetic algorithm, or a reinforcement value used by a learning algorithm. These values are usually calculated by an evaluation function designed by the SAB researcher to favor behaviors that achieve a particular goal or task. Here, we propose a radically different approach to evaluations, based on imitating successful behaviors, rather than evolving or inventing them from scratch. One could potentially use human-generated trajectories as training exemplars for teaching animats what to do in particular situations. This can be more efficient than other methods of direct human reinforcement [Nehmzow and McGonigle, 1994], in which robot designers must hover over their creations, giving them positive and negative reinforcements every time they do something. Instead, one can develop error measures based on deviations between the animat's behavior and the trajectory generated by a human subject. Of course, we generally want to capture the overall patterns of motion in some category rather than the specificities of any one human-generated trajectory, so the error measures will have to be developed accordingly.

5.4 Extracting cues of animate motion through iterative experiments

We propose a simple method to help identify the basic cues of animacy by having human subjects play the fundamental motion games that all animal species play, in a much simplified computer form. The multi-step research method that lies at the heart of our research program proceeds as follows: (1) collect human-generated motion data for representative tasks; (2) develop useful measures that bring out the important category-distinguishing parameters in that data (this is the hard part!); (3) vary those parameters in animats to generate new kinds of motion trajectories; and (4) have human subjects judge those new trajectories as inanimate, nicely animate, weirdly animate, or super-animate in controlled experiments. These steps will be iterated as necessary until we know how to generate animacy "super-stimuli," indicating that we have identified some important animacy cues. We have made progress in the first three steps of this procedure, as described in the previous sections, but the complete iterative process remains to be fulfilled.

6 Conclusions

Our extraordinary human perceptual capacities for recognizing and classifying animate motion have been a mixed blessing for the behavioral sciences and for SAB research. On the one hand, they make animal field studies and observational psychology easy: we just watch what the chimp does and can usually decide whether she is chasing, escaping, courting, or playing. On the other hand, we not only anthropomorphize animals, we also attribute animacy and intentionality to almost anything to moves, and yet rarely bother to ask how we achieve such fast, effortless, accurate feats of perception.

Our intuitive assessments of animacy are not sufficient. We need to be able to compare motion patterns generated by our animats to motions generated by real organisms. Because other animals don't follow directions very well or interact with computer simulations very skillfully, we have explored using human subjects to generate benchmark motion data that is directly comparable to animat-generated data. Even the simple measures and analyses reported here can distinguish among different classes of behaviors, and allow informative comparisons against the behavior of simple simulated control systems. In addition, we have demonstrated the use of a motion-based Turing test as a method for evaluating animate robot algorithms. We have also reported four applications of this human-simulation approach in SAB research, suggesting that whenever we set our animats some task in some environment, we should try out the task ourselves first, to see how we fare. In the great tradition of psychopharmacologists who always try their own drugs before selling them to others, we should never force a virtual environment on our animats that we haven't first tasted ourselves.

References

[Andersson, 1994] Andersson, M. (1994). *Sexual Selection.* Princeton University Press.

[Beer, 1995] Beer, R. D. (1995). A dynamical systems perspective on agent-environment interaction. *Artificial Intelligence*, 72:173–215.

[Blumberg, 1994] Blumberg, B. (1994). Action-selection in hamsterdam: Lessons from ethology. In Cliff, D., Husbands, P., Meyer, J. A., and Wilson, S., editors, *From Animals to Animats 3*, pages 108–117. Cambridge: MIT Press.

[Cliff et al., 1993] Cliff, D., Harvey, I., and Husbands, P. (1993). Explorations in evolutionary robotics. *Adaptive Behavior*, 2:73–110.

[Cutting and Kozlowski, 1977] Cutting, J. and Kozlowski, L. (1977). Recognition of friends by their walk. *Bulletin of the Psychonomic Society*, 9:353–356.

[Dittrich, 1993] Dittrich, W. (1993). Action categories and the perception of biological motion. *Perception*, 22:15–22.

[Driver and Humphries, 1988] Driver, P. M. and Humphries, N. (1988). *Protean behavior: The biology of unpredictability.* Oxford: Oxford University Press.

[Fagen, 1981] Fagen, R. (1981). *Animal Play Behavior.* New York: Oxford University Press.

[Gelman, 1990] Gelman, R. (1990). First principles organize attention to and learning about relevant data: number and the animate-inanimate distinction as examples. *Cognitive Science*, 14:79–106.

[Johansson, 1975] Johansson, G. (1975). Visual motion perception. *Scientific American*, 232:76–88.

[Mather and West, 1993] Mather, G. and West, S. (1993). Recognition of animal locomotion from dynamic point-light displays. *Perception*, 22:759–766.

[May, 1991] May, M. (1991). Aerial defense tactics of flying insects. *American Scientist*, 79:316–328.

[Miller and Freyd, 1993] Miller, G. and Freyd, J. (1993). The interplay among evolutionary cognitive and behavioral dynamics. Technical Report CSRP 290, University of Sussex.

[Miller, 1996] Miller, G. F. (1996). Sexual selection in human evolution: Review and prospects. In Crawford, C. and Krebs, D., editors, *Evolution and Human Behavior: Ideas, Issues, and Applications.* Lawrence Erlbaum.

[Miller and Cliff, 1994] Miller, G. F. and Cliff, D. (1994). Protean behavior in dynamic games: Arguments for the coevolution of pursuit-evasion tactics in simulated robots. In Cliff, D., Husbands, P., Meyer, J. A., and Wilson, S., editors, *From Animals to Animats 3*, pages 411–420.

[Nehmzow and McGonigle, 1994] Nehmzow, U. and McGonigle, B. (1994). Achieving rapid adaptations in robots by means of external tuition. In Cliff, D., Husbands, P., Meyer, J. A., and Wilson, S., editors, *From Animals to Animats 3*, pages 301–308. Cambridge: MIT Press.

[Oram and Perrett, 1994] Oram, M. and Perrett, D. (1994). Responses of anterior superior temporal polysensory (stpa) neurons to biological motion stimuli. *Journal of Cognitive Neuroscience*, 6(2):99–116.

[Port and van Gelder, 1995] Port, R. F. and van Gelder, T. (1995). *Mind as motion: Explorations in the dynamics of cognition.* Cambridge: MIT Press.

[Premack, 1990] Premack, D. (1990). The infant's theory of self-propelled objects. *Cognition*, 36:1–16.

[Reynolds, 1994] Reynolds, C. W. (1994). Competition, coevolution and the game of tag. In Brooks, R. A. and Maes, P., editors, *Artificial Life IV*, pages 59–69. Cambridge: MIT Press.

[Terzopoulos et al., 1994] Terzopoulos, D., Tu, X., and Grzeszczuk, R. (1994). Artificial fishes with autonomous locomotion, perception, behavior, and learning in a simulated physical world. In Brooks, R. A. and Maes, P., editors, *Artificial Life IV*, pages 17–27. Cambridge: MIT Press.

[Todd and Wilson, 1993] Todd, P. M. and Wilson, S. W. (1993). Environmental structure and adaptive behavior from the ground up. In Meyer, J. A., Roitblat, H. L., and Wilson, S. W., editors, *From Animals to Animats 2*, pages 11–20.

The Engineering of Mind

James S. Albus
Intelligent Systems Division
National Institute of Standards and Technology
Gaithersburg, MD 20895
albus@cme.nist.gov

Abstract

While the mind remains a mysterious and inaccessible phenomenon, many of the components of mind, such as perception, behavior generation, knowledge representation, value judgment, reason, intention, emotion, memory, imagination, recognition, learning, attention, and intelligence are becoming well defined and amenable to analysis. Progress is rapid in the cognitive and neurosciences as well as in artificial intelligence, control theory, and many other fields related to the engineering of mind. A reference model architecture for intelligent systems is suggested to tie together concepts from all these separate fields into a unified framework that includes both biological and machine embodiments of the components of mind. It is argued that such a reference model architecture will facilitate the development of scientific models of mind.

1. Introduction

What is mind? What is the relationship between the mind and the brain? What is thought? What are the mechanisms that give rise to imagination? What is perception and how is it related to the object perceived? What are emotions and why to we have them? What is will and how do we choose what we intend to do? How do we convert intention into action? How do we plan and how do we know what to expect from the future?

Until recently such questions could only be addressed indirectly by subjective introspection, or by psychological experiments in which the majority of the critical variables cannot be measured or controlled. Only in the past half century, since the invention of the electronic computer has it become possible to approach these issues directly by building machines and programs that exhibit some of the mind's essential qualities; such as the ability to recognize patterns and relationships, to store and use knowledge, to reason and plan, to learn from experience, and to evaluate what is happening. This is a crucial step in the study of mind, for it makes it possible to build mathematical models, and conduct experiments where, at least in principle, all the variables can be measured.

Research in neural nets, brain modeling, fuzzy systems, and genetic algorithms is providing insight into learning and the similarities and differences between neuronal and electronic computing. Artificial intelligence and linguistics are probing the nature of language. Image understanding has developed into a field of its own. There has been significant progress in rule based reasoning, planning, and problem solving. Game theory and operations research have developed methods for decision making in the face of uncertainty. Autonomous vehicle research has produced advances in real-time sensory processing, world modeling, navigation, and locomotion. Research in robotics and automated manufacturing has produced intelligent hierarchical controls, distributed databases, representations of object geometry, process plans, and material properties. Control theory has developed precise understanding of stability, adaptability, and controllability under various conditions of feedback and noise. Powerful mechanisms have been developed for parallel processing, recursive estimation, and focusing of attention. Engineering solutions exist for fusing sensory input from multiple sources, and assessing the believability of noisy data.

Since the 1950's, a wide variety of robotic systems have been designed and built -- from experimental laboratory vehicles that wander about, follow walls, and pick up sundry items, to precision assembly systems that use vision to acquire parts and Computer Aided Design (CAD) models to plan motions. Many approaches have been explored and various architectures designed, from subsumption and neural nets, to SOAR [Rosenbloom et al. 93] and RCS [Albus 96]. Entire factories have been automated and products as complex as the Boeing 777 aircraft containing over three million parts have been designed and engineered entirely in software, without physical mockups.

Progress is also rapid in the cognitive and neurosciences. Neuroanatomy is producing maps of the interconnecting pathways of the brain. Neurophysiology is determining how neurons compute functions and communicate information. Neuropharmacology is discovering many of the transmitter substances that modify value judgments, compute reward and punishment, motivate behavior, and produce learning. Psychophysics provides clues as to how humans and animals perceive objects, events, time, and space, and reason about relationships. Behavioral

psychology is creating models of mental and emotional development and behavior.

While the mind itself remains a mysterious and inaccessible phenomena, many of the components of mind, such as perception, behavior generation, knowledge representation, value judgment, reason, intention, emotion, memory imagination, recognition, learning, and intelligence are becoming well defined and amenable to analysis. Progress is rapid, and there exists an enormous and rapidly growing literature in each of the above fields. What is lacking is a general theoretical model which ties all these separate areas into a unified framework that includes both biological and machine embodiments of the components of mind. In 1991, I published an outline for a general theory of intelligence [Albus 91]. This theory was expressed in the notation of the Real-time Control System (RCS) developed at NIST and elsewhere for the design of intelligent control systems [Albus 96]. In this paper, I will illustrate how many of the concepts developed for intelligent machines apply to biological intelligence and suggest how engineering principles might be developed for the design and analysis of practical intelligent systems.

2. The Fundamental Elements

In any scientific endeavor, it is necessary to precisely define concepts and clearly state assumptions. I therefore begin with some axioms and definitions.

Axiom 1: The functional elements of an intelligent system are behavior generation, sensory perception, world modeling, and value judgment.

Df: **behavior generation (BG)**
the planning and control of action designed to achieve behavioral goals.

Behavior generation organizes the response of a collection of agents to task commands. Behavior generation accepts task commands with goals, objects, and priorities. It decomposes tasks into jobs and assigns jobs and resources to agents. It formulates and selects plans and develops schedules for possibly coordinated actions by agents. It executes plans and reacts to feedback so as to follow plans in spite of local perturbations and unexpected events. Finally, behavior generation produces output commands that are either decomposed further, or act directly on the environment.

Df: **planning**
a process that:

1. assigns responsibility to agents for jobs, and allocates resources to agents for performing their assigned jobs,
2. hypothesizes strings of actions (plans) for agents from a vocabulary of possible actions to accomplish jobs,
3. simulates and predicts the results of executing these hypothesized plans,
4. evaluates the predicted results of the hypothesized plans,
5. selects the hypothesized plan with the most favorable results for execution.

The planning process may use either a heuristic or an exhaustive search strategy for synthesizing hypothesized plans. Heuristic strategies may include the selection of previously generated plans from a library.

Df: **agent**
a set of computational elements that plan and control the execution of jobs, correcting for errors and perturbations along the way.

An agent may servo its output to follow a planned trajectory, or may sequence discrete actions and branch on conditions. An agent also assigns jobs and resources to subordinates. The computational elements in an agent may include sensory perception, world modeling, and value judgment functions and a knowledge database.

Df: **sensory perception (SP)**
the transformation of data from sensors into meaningful and useful representations of the world.

Sensory perception accepts input data from sensors that measure states of the external world as well as internal states of the system itself. Sensory perception scales and filters data, computes observed features and attributes, and compares observations with expectations generated from internal models. Correlations between sensed observations and internally generated expectations are used to detect events and recognize entities and situations. Differences between sensed observations and internally generated expectations are sent to world modeling to update internal models. Sensory perception also classifies, generalizes, and clusters, or groups, recognized entities and detected events into higher order entities and events, and computes attributes of entities and events.

Df: **value judgment (VJ)**

a) the computation of cost, risk, and benefit of actions and plans,
b) the estimation of the importance and value of objects, events, and situations,
c) the assessment of reliability of information,
d) the calculation of reward or punishment resulting from perceived states and events.

Value judgment evaluates perceived and planned situations thereby enabling behavior generation to select advantageous goals and set priorities among competing behavioral possibilities. It computes what is important (for attention), and what is rewarding or punishing (for learning). Value judgment is performed in the brain by the limbic system.

Df: **world modeling (WM)**
a process that performs four principal functions:

1. Uses sensory input to construct, update, and maintain a knowledge database, including iconic images, symbolic lists, entity and event frames, and semantic and pragmatic relationships between entities, and links between symbolic and iconic representations. In biological systems, this is the function of short term and long term memory.
2. Answers queries from behavior generation regarding the state of the world. It provides knowledge about the state of the world to be used by behavior generation as feedback to servo behavior to follow desired plans, and to provide the latest status information on which to base planning operations. This is the function of recall.
3. Simulates results of possible future plans. Simulated results are evaluated by the value judgment system in order to select the best plan for execution. Simulation performed by the brain is what we call thinking. Our ability to plan depends on the fidelity of our internal model of how the world works.
4. Generates sensory expectations based on knowledge in the knowledge database. Expectations are used by sensory perception to configure filters, masks, windows, and templates for correlation, model matching, and recursive estimation; and for clustering. This corresponds to the tendency of biological brains to see and hear what they expect to see and hear.

Axiom 2: The functional elements of an intelligent system are supported by a knowledge database that stores a priori and dynamic information about the world in the form of state variables, symbolic entities, symbolic events, rules and equations, structural and dynamic models, task knowledge, signals, images, and maps.

Df: **knowledge database (KD)**
a set of data structures filled with the static and dynamic information that provide a best estimate of the state of the world and the processes and relationships that effect events in the world.

In the knowledge database:

State variables represent the current estimated state of the world.

Entity frames are list data structures that store symbolic representations of features, objects, or groups that exist in the world, or in the imagination. An entity frame consists of a list head with a name as an address, plus a set of attribute-value pairs, and a set of relations to other entities or events. These relationships represent semantic meaning.

Event frames are list data structures that store symbolic representations of state transitions, or situations that occur at particular times and places, or sequences of states or situations that transpire over intervals of time and space in the world. An event frame also consists of a name, a set of attribute-value pairs, and a set of relationships to other events or entities.

Rules and equations such as if/then rules, the predicate calculus, and differential equations can express physical laws that describe the way the world works, as well as mathematical and logical formulae that describe the way things relate to each other.

Images are two-dimensional functions of attribute values that may be sampled by arrays of sensors, or pixels. Images may be generated in a number of ways. For example, an image may be formed by the optical projection of light from a scene in the world through a lens onto an array of photoreceptors (or pixels) such as the retina or a CCD TV camera. An image may also be formed by pressure on an array of tactile sensors on the skin. An image consists of attributes such as brightness, pressure, spatial or temporal gradients, stereo disparity, or computed values such as range, or flow that are derived from other images. An image may also be generated by internal mechanisms (such as a computer graphics engine) from information stored in symbolic entity frames. In biological systems, image generation corresponds to imagination [Kosslyn 94].

Maps are also two-dimensional arrays of pixels, wherein icons or symbolic names, in addition to attributes, are attached to pixels.

Task knowledge is knowledge of how to do things. Task knowledge includes information about the goal, the agents, the task objects, parameters, enabling and disabling conditions, tools and resources required, and plans, scripts, or procedures for generating and refining plans. Control laws and plant models can also be represented as task knowledge.

The knowledge database has two parts: long term and short term memory.

a) *Long term* (static or slowly varying) memory contains symbolic representations of all the entities, events, and rules that are known to the intelligent system. Long term memory consists entirely of symbolic entity and event frames, plus rules and equations. Attributes from long term frame representations may be transferred into short term memory, or vice versa.

b) *Short term* (dynamic) memory contains both symbolic and iconic representations of entities-of-attention.

Df: entities-of-attention
entities that have either been specified by the current task, or are particularly noteworthy entities observed in current sensory input.

Short term symbolic entity frames include attributes, pointers to iconic images, and pointers to entities stored in long term memory. Short term iconic representations can consist of attribute images generated directly from sensory observations, or filtered through recursive estimation. Short term iconic images can also be generated by internal

mechanisms from short term symbolic entity frames. In machine systems, this is done through simulation and animation. In biological systems, it corresponds to imagination. Short term iconic images can be used to mask or window incoming data, or to compare and fuse incoming sensory observations with internally generated images. Short term iconic images persist in memory only so long as they are refreshed by incoming sensory data or by internally generated images.

Axiom 3: The functional elements and knowledge database can be implemented by a set of computational modules that are interconnected to form nodes in a control system architecture.

Df: **node**
a part of a control system that processes sensory information, maintains a world model, computes values, and generates behavior.

A node corresponds to a set of neurons in the brain that close a loop between afferent and efferent neural pathways. In doing so, each node typically performs the functions of behavior generation, sensory perception, world modeling, and value judgment. A typical node from the RCS reference model architecture [Albus 96] is shown in Figure 1.

Within each node, interconnections between behavior generation, world modeling, and value judgment modules enable task decomposition, planning, and reasoning about future actions. Interconnections between sensory perception, world modeling, and value judgment modules enable knowledge acquisition, situation evaluation, and learning. Interactions between sensory perception and world modeling modules enable recursive estimation for optimal filtering and prediction. Interconnections between sensory perception, world modeling, and behavior generation modules close a reactive feedback control loop between the observed input and the commanded action. Input commands convey task goals and priorities from higher level nodes. Output commands convey subtask goals to lower level nodes, or directly to actuators. The downward flow of commands corresponds to efferent pathways in the brain. The upward flow of information through the sensory perception modules correspond to afferent pathways.

Connections to the operator interface have no biological analog. These enable a human operator to insert commands to override or modify system behavior, or to observe the values of state variables and entity attributes. The operator interface can also be used for system maintenance, programming, and debugging.

Axiom 4: The complexity inherent in intelligent systems can be managed through hierarchical layering.

Intelligent systems are inherently complex. Hierarchical layering is a common method for organizing complex systems that has been used in many different types of organizations throughout history for effectiveness and efficiency of command and control. In a hierarchical control system, higher level nodes have broader scope and longer time horizons, with less concern for detail. Lower level nodes have narrower scope and shorter time horizons, with more focus on detail.

In the RCS reference architecture, behavior generating modules in nodes at the upper levels in the hierarchy make long range strategic plans consisting of major milestones, while lower level behavior generating modules successively refine the long range plans into short term tactical plans with detailed activity goals. Sensory perception modules at lower levels process data over local neighborhoods and short time intervals, while at higher levels, they integrate data over longer time intervals and larger spatial regions.

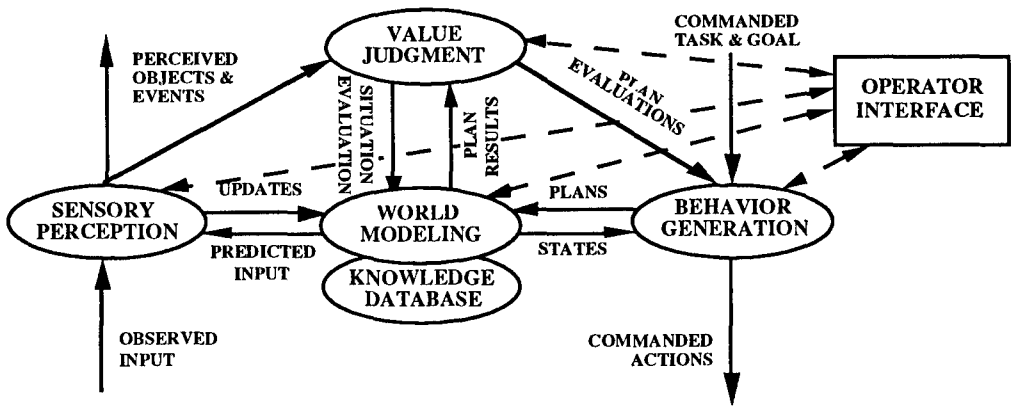


Figure 1. A node in the RCS reference model architecture. The functional elements of an intelligent system are behavior generation (planning and control), sensory perception (filtering, detection, recognition, and interpretation), world modeling (storing and retrieving knowledge and predicting future states), and value judgment (computing cost, benefit, importance, and uncertainty). These are supported by a knowledge database, and a system architecture that interconnects the functional modules and the knowledge database. This collection of modules and their interconnections makes up a generic node in the RCS reference model architecture. Each module in a node may have an operator interface.

World model knowledge at low levels, is short term and fine grained, while at higher levels it is broad in scope and generalized. At every level, feedback loops are closed to provide reactive behavior, with high-bandwidth fast-response loops at lower levels, and slower more deliberative reactions at higher levels. RCS thus provides what Brooks calls "coherent behavior from many adaptive processes" [Brooks 94].

At each level, state variables, entities, events, and maps are maintained at the resolution in space and time that is appropriate to that level. At each successively lower level in the hierarchy, as detail is geometrically increased, the range of computation is geometrically decreased. As temporal resolution is increased, the span of interest decreases. As plans become more detailed, the planning horizon shrinks. This produces a ratio that remains relatively constant throughout the hierarchy. A design goal is for behavior generating functions at each level to generate plans of roughly the same number of steps, and for sensory perception functions to compute entities containing roughly the same number of subentities. At higher levels, plans, perceived entities, and world modeling simulations are more complex, but there is more time available between replanning intervals for processes to run. Thus, hierarchical layering keeps the amount of computing resources needed in each node within manageable limits.

At the top of the hierarchy, strategic objectives and priorities influence the selection of goals and the prioritization of tasks throughout the entire hierarchy. However, the details of execution are left to subordinates.

At intermediate levels, tasks with goals and priorities are received from the level above, and subtasks with subgoals and shorter range attention priorities are output to the level below. Again, the details of execution are left to subordinates.

At each level, global goals from a higher level are refined and focused onto more narrow and finer resolution subgoals. At each level, attention is focused into a more narrow and finer resolution view of the world. The effect of each hierarchical level is thus to geometrically refine the detail of the task and the view of the world, while keeping the computational load at all levels within limits that can be handled by intelligent agents of modest capacity.

3. The RCS Reference Model Architecture

A set of generic nodes such as illustrated in Figure 1 can be interconnected in a hierarchical control architecture for an intelligent machine system. The diagram shown in Figure 2 consists of a hierarchy of control nodes wherein each of the nodes contain the set of modules and interconnections illustrated in Figure 1. Each of the nodes is therefore an intelligent controller capable of planning and control, knowledge representation, value judgment, and sensory perception. Each of the nodes closes a feedback loop between afferent and efferent pathways. An operator interface may access modules in any node at any level.

The example of a reference model architecture in Figure 2 is designed for a military vehicle system with four subsystems: locomotion, mission package, communication, and attention. Each of the four subsystems has one or more mechanisms, each of which has one or more actuators and sensors. For example, the locomotion subsystem may consist of a navigation and driving controller with several controllers for steering, braking, throttle, and gear shift, plus ignition, lights, horn, and turn signals, each of which has one or more actuators and sensors. The mission package may have loading, aiming, and firing subsystems each of which has numerous sensors and actuators. The communication subsystem might consist of a message encoding subsystem, a protocol syntax generator, and communications bus interface, plus antenna pointing and band selection actuators. The attention subsystem may contain cameras, laser range imagers, infrared cameras, radar, and acoustic sensors. Sensory perception algorithms may detect and track objects, surfaces, edges and points, and compute trajectories for pan, tilt, and focus actuators to point cameras, range finders, and antennae. All of these functions need to be coordinated in order to successfully achieve behavioral goals. The horizontal curved lines between WM modules represent the sharing of state information between nodes within subtrees in order to synchronize related tasks.

The operator interface provides the capability for the operator to interact with the system at any time at a number of different levels—to adjust parameters, to change speed, to select or verify targets, or to authorize the use of weapons. The operator interface provides a means to insert commands, change missions, halt the system, alter priorities, perform identification of friend-or-foe, or monitor any of the system functions. The operator interface can send or display information from the communications subsystem, or display any of the state variables in the world model at a rate and latency dictated by the communications bandwidth. Using the operator interface, a human operator can view situational maps with topographic features, with overlays that indicate the position and movement of both friendly and enemy forces. The operator interface may display graphic images of motion paths, or print out control programs (plans), in advance, or while they are being executed. The operator interface also enables the operator to run diagnostics in the case of system malfunctions.

In Figure 2, three levels of control are shown above the node representing the individual vehicle. These three additional levels represent a virtual chain of command that exists above the individual vehicle. Because each vehicle is semi-autonomous, it carries a copy of the control nodes that contain its superiors in the command chain. This virtual chain of command serves four functions. First, it provides each vehicle with an estimate of what its superiors would command it to do if they were in direct communication. Second, it enables, any vehicle to assume the duties of any

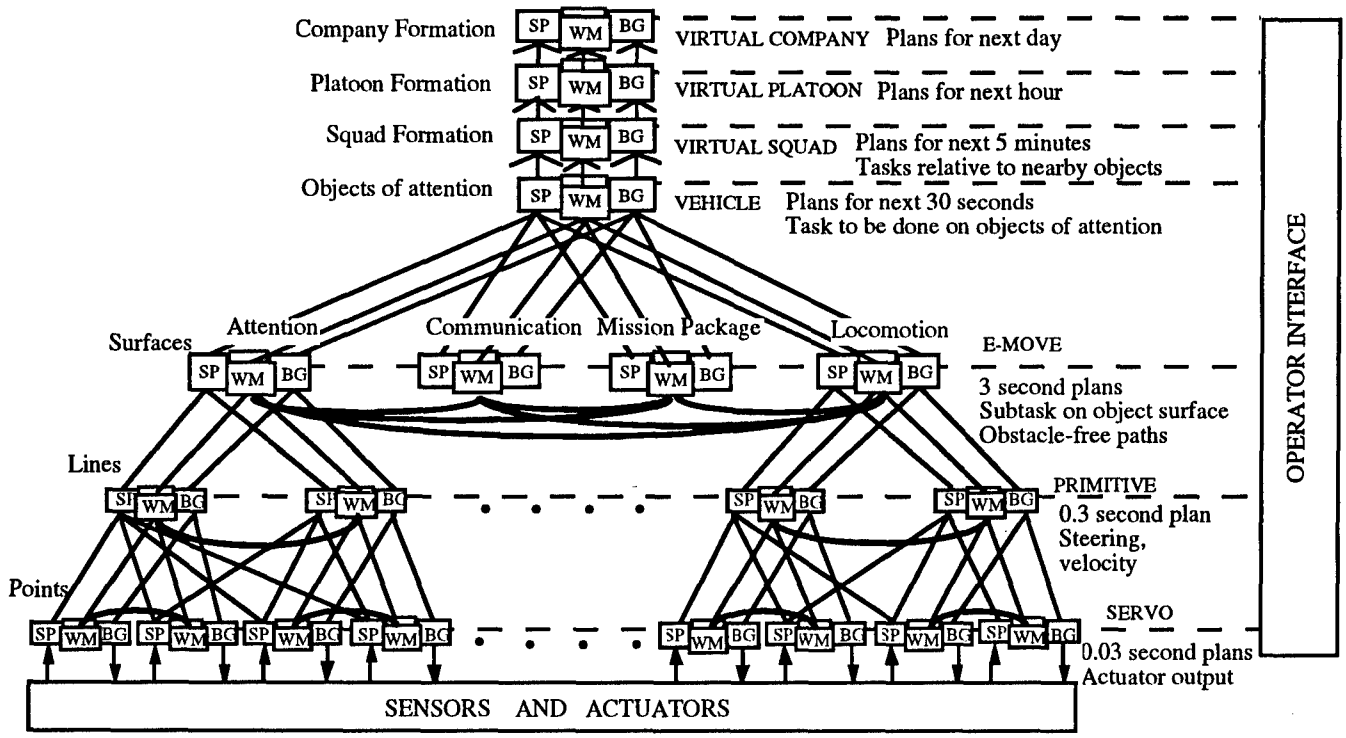


Figure 2. A RCS reference model architecture for an individual vehicle. Processing nodes are organized such that the behavior generation (BG) modules form a command tree. Information in the knowledge database (KD) is shared between world modeling (WM) modules in nodes within the same subtree. KD modules are not shown in this figure. On the right, are examples of the functional characteristics of the behavior generation (BG) modules at each level. On the left, are examples of the type of entities recognized by the sensory perception (SP) modules and stored by the WM in the KD knowledge database at each level. Sensory data paths flowing up the hierarchy typically form a graph, not a tree. Value judgment (VJ) modules are hidden behind WM modules. A control loop is closed at every node. An operator interface may provide input to, and output from, modules in every node.

of its superiors in the event this should become necessary. Third, it enables each vehicle to dedicate a separate node to handle higher level tasks. In this example, the virtual chain of command consists of three levels with three different planning horizons (five minutes, one hour, and one day). These three levels deal with external objects and maps at three different scales and ranges. Fourth, it provides a natural interface for human commanders at the squad, platoon, or company level to interface with the vehicle at a level relevant to the task being addressed. There, of course, may be more than three levels above the vehicle in the virtual chain of command.

At each layer of the RCS hierarchy, there are both deliberative and reflexive elements. In each node at each level, sensory data are processed, entities are recognized, world model representations are maintained, and tasks are deliberatively decomposed into parallel and sequential subtasks, to be performed by cooperating sets of subordinate agents. Also in each node, feedback from sensors reflexively closes a control loop allowing each agent to respond and react to unexpected events. The result is a system that combines and distributes deliberative and reflexive features throughout the entire hierarchical architecture, with both planned and reactive capabilities tightly integrated at all levels and time frames. Figure 3 illustrates the internal structure of a behavior generation (BG) module. The BG module represents an operational unit that typically is comprised of several intelligent agents. It is important to distinguish clearly between the agents and the BG organizational unit to which they belong. An agent is typically part of two BG modules at two different levels. At one level, an agent is a member of a team, and subordinate to the supervisor of the team. At the next lower level, the same agent is a supervisor of a team of agents in the BG module at the next lower level. The job assignor submodule thus is part of the supervisor agent for a BG unit at level i. The job assignor assigns jobs to agents in the BG unit, and works with them to develop a set of schedules for each of the subordinate agents within the i-level BG unit. The set of schedules represents a plan for the BG unit. Each of the subordinate agents in the i-level BG unit contains a scheduler and an executor at level i, and is a supervisor (i.e.,

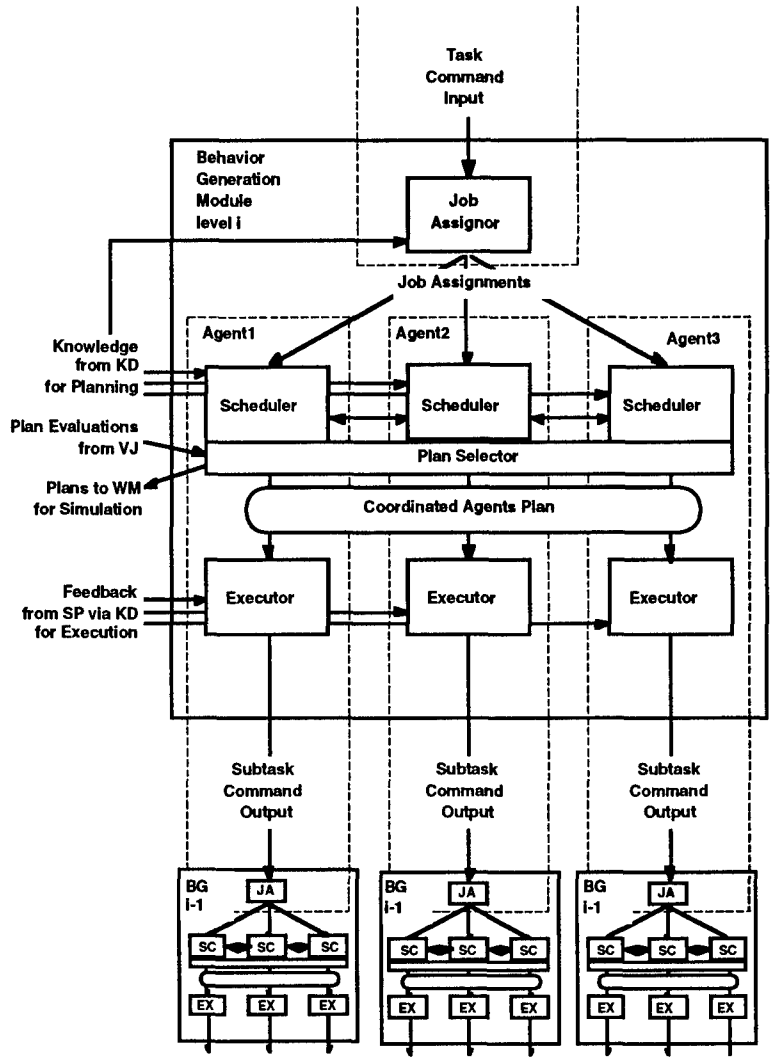


Figure 3. Internal structure of a Behavior Generation (BG) module. A BG module is an organizational unit consisting of a Job Assignor, a set of Schedulers, a Plan Selector, a plan holding register that contains the coordinated agents plan, and a set of Executors. The Schedulers, Executors, and Job Assignors comprise agents within the BG units. Each agent is part of two BG units at two different levels. At level i, an agent is a team member, or peer. At level i-1, the same agent is a supervisor.

a job assignor) at level i-1. Each agent is thus a peer in a BG module at a level, and a supervisor of a BG module at the next lower level.

Figure 4 illustrates the details of the interactions that take place within and between the BG, WM, SP, and VJ modules in a single node. A task command into a node is of the form <Do action on object to achieve goal x*>. The task name and goal enters a BG module where the task is decomposed into jobs for agents. Each agent has a scheduler that generates a schedule and coordinates with schedulers of other agents. This produces a tentative plan which is a path from the current estimated state $\hat{x}$ to the goal state x*. The tentative plan is submitted to a plan simulator in the WM which predicts results. These results are evaluated by the plan evaluator in the VJ which returns cost-benefit analysis to the plan selector. If the evaluation is satisfactory, the tentative plan is selected for execution. Otherwise replanning is called for, and another tentative plan is generated by the job assignor and scheduler submodules.

The object specified by the task command is sent to the world model knowledge database which looks up a corresponding entity in the long-term memory database. This entity is then entered into the short-term memory list of entities-of-attention. Attributes of entities-of-attention are used to generate a predicted image that can be compared with an observed image. The predicted image defines windows for correlation, comparison, recognition, and clustering. Differences between predicted and observed

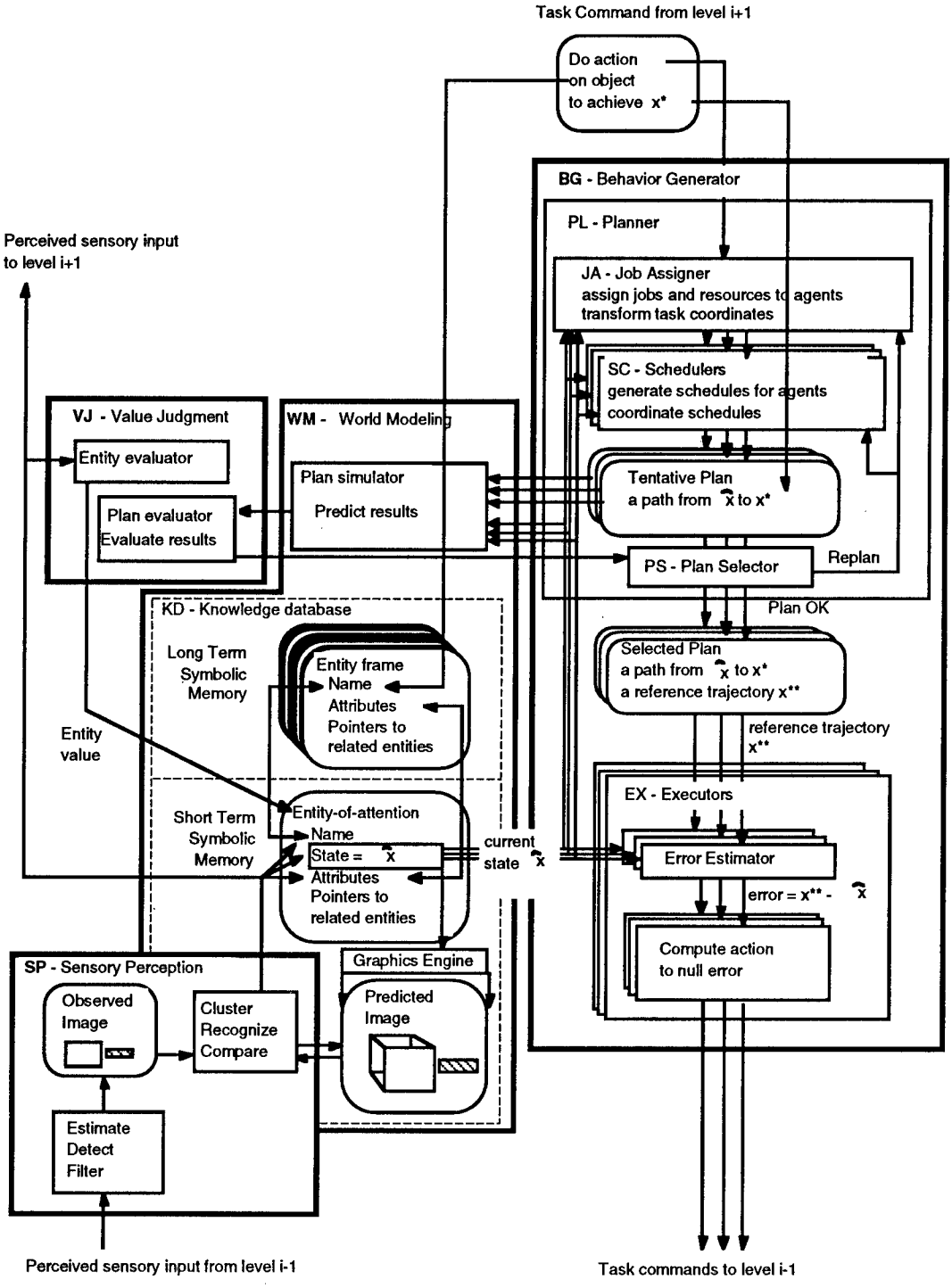


Figure 4. Relationships within a single node of the RCS architecture. The behavior generating (BG) modules contain job assignor (JA), scheduler (SC), plan selector (PS) and executor (EX) submodules. The world modeling (WM) module contains a plan simulator and mechanisms for updating the knowledge database (KD), which contains both long term and short term symbolic representations and short term iconic images. The sensory perception (SP) module contains filtering, detecting, and estimating algorithms, plus mechanisms for comparing predictions generated by the WM module with perceived input from sensors. It has algorithms for recognizing entities and clustering entities into higher-level entities. The value judgment (VJ) module evaluates plans and places values on entities recognized in the observed sensory input.

images can be used to update the estimated state $\hat{x}$, including the estimated attributes of the entity-of-attention. This is a looping process of recursive estimation that can be used to implement a tracking filter, a predictor, or a phase-lock loop. The estimated state $\hat{x}$ is used by the executor to servo the output to follow the selected plan. It is also used by the job assignor and scheduler submodules for planning, and by the WM plan simulator for predicting results of tentative plans. Recognized and clustered entities are evaluated by the VJ and transmitted to higher level sensory perception modules for more global processing.

This recursive estimation procedure has recently been implemented on modest computing hardware for vision based lane following by highway vehicles. Dickmanns [92] has implemented a 4-D model in world coordinates. Schneiderman and Nashman [94] implemented a 2-D model in image coordinates. Speeds of up to 100 kilometers per hour on highways and 40 kilometers per hour on a winding test track have been achieved by the 2-D system. The 4-D system has achieved speeds of up to 130 kilometers per hour in normal highway traffic with automatic lane changing while simultaneously tracking up to five other vehicles. [Dickmanns 95] **Axiom 5:** The complexity of the real world environment can be managed through focusing attention.

Intelligent systems must operate in a real world environment which is infinitely rich with detail. The real world environment contains an unlimited variety of objects, such as ground, rocks, grass, sand, mud, trees, bushes, buildings, posts, ravines, rivers, roads, vehicles, weapons, people, and enemy and friendly positions. The environment also contains elements of nature, such as wind, rain, snow, sunlight, and darkness. All of these objects and elements have an infinite regression of detail, and the world itself extends infinitely far in every direction.

Yet, the computational resources available to any intelligent system are finite. No matter how fast and powerful computers become, the amount of computational resources that can be embedded in any practical system will be limited. Therefore, it is imperative that the intelligent system be able to focus the available computing resources on what is important, and ignore what is irrelevant.

Top down, what is important is defined by behavioral goals. Top down goals and high level perceptions generate expectations of what should be encountered during the evolution of the task. Bottom up, what is important is the unexpected, unexplained, unusual, or dangerous. The lower level sensory perception functions detect variations between what is expected and what is observed. The lower levels also compute attributes of signals or images that may indicate problems or emergency conditions, such as limits being exceeded on position, velocity, acceleration, vibration, pressure, force, current, voltage, or thermal sensor signals.

Focusing of attention can be accomplished by sampling the environment at high resolution at important points, and with low resolution elsewhere. For example, most of the photodetectors in the visual field are concentrated in the foveal region, and lower resolution in the periphery. Similarly, tactile attention is accomplished by high resolution spatial distribution of tactile sensors in the finger tips, lips, and tongue, with much lower resolution in other regions of the skin. Intelligent control points the fovea at points in the visual field that are important, and guides the fingers to touch objects at points important to the task goal. Hierarchical layering focuses computing resources near the present, with exponentially lower resolution for longer time horizons, both for planning the future and analyzing the past.

5. Discussion

In RCS, each node, and each module and submodule within a node, is implemented as an augmented finite state machine which runs asynchronously as a cyclically executing process. The execution cycle typically is triggered by a clock at a fixed repetition rate that is an order of magnitude faster than the time constant of the process being controlled, but may be triggered by events. Each finite-state machine is surrounded by a set of input and output data buffers. At the beginning of each cycle, the submodule reads from its input buffers and processes the inputs into a form suitable for a state-table that encodes a set of state dependent if/then rules of a form that are common in expert systems. The processed input is compared with the rules in the state-table. The rule that matches causes the process to go to a next state, possibly execute a procedure, and compute an appropriate output. Each submodule also computes a set of diagnostic functions, such as the time required for the process to complete, the maximum time the process has taken, and the average time taken. Each submodule has an interface for an operator that provides the operator the ability to halt, single-step, and/or display the value of any variable and the state of any process at any time during execution. A process may be halted, parameters examined by a programmer, variables changed, and execution resumed. Communications between processes in the RCS system are designed so that all processes cycle independently and run completely asynchronously with no protection against messages getting overwritten. In this respect, RCS is similar to Brooks' $\pi\beta$ machine. [Brooks 94] We agree with Brooks that design principles and computational structures should constrain system design and force solutions to maintain biological relevance. To do otherwise allows divergence from approaches that can draw on (or contribute to) concepts from the cognitive and neurosciences. Our differences with Brooks include our insistence on a systematic design of the topology of the computing structure. While we admit that random design choices coupled with natural selection may eventually lead to an optimum topology, we believe that systematic design principles will reach the goal in far fewer iterations.

RCS programming tools and software templates are being built that provide the system developer an easy way to configure an RCS system. The templates automatically generate all the required utilities and diagnostic features, and provides slots in a menu for inputs, outputs, and system

parameters. Software templates are implemented in C++. A graphical design tool enables a programmer to define a RCS submodule with the click of a mouse, and interconnect submodules by click and draw techniques. These programming tools have enabled RCS systems with hundreds of submodules to be designed and built in a few months.

RCS has been implemented on a variety of platforms, including Sun workstations, 486 and Pentium class PC computers, VME systems, and Macintosh machines using a number of different operating systems, including Forth, pSOS, DOS, VxWorks, and Lynx OS[1]. The overhead for a RCS template running on a 486 class machine is about five microseconds. The cycle time for a typical low level RCS submodule is 30 milliseconds. However, for high performance machine tool applications, servo loops may be closed every 300 microseconds.

6. Conclusions

While many deep theoretical issues regarding the nature of the mind remain, much is known and progress is rapid. Intelligent machines research is beginning to yield to engineering approaches. Intelligent systems are beginning to exhibit impressive performance capabilities in practical applications such as manufacturing systems and highway vehicles. Some of the most promising lines of research appear to be in:

a) combining goal-driven planning with reflexive behavior,

b) focusing of attention and active control of sensing,

c) hierarchical decomposition of tasks and goals,

d) the use of recursive estimation for sensory perception,

e) the development of a reference model architecture for intelligent system design.

A reference model architecture paves the way for design principles and software engineering tools that facilitate the building of intelligent machine systems.

Recent work in open system architectures for intelligent controllers is leading toward specification of canonical functional modules and standards for application programming interfaces (APIs) between functional modules in open architecture machine controllers. [TEAM 96] API standards promise to facilitate system integration and enable incremental upgrades of capability.

It soon may be possible to add new layers or subsystems without modifying system software. Eventually, systems developed in different laboratories might be integrated into large experimental systems consisting of thousands of computing platforms in hundreds of different laboratories. At that point, intelligent machine systems might approach the complexity and computing power of the human brain. Under such circumstances, the prospect is bright for building scientifically valid experimental models of the mind.

It should be noted, in closing, that such models would not only advance the scientific inquiry into of the nature of mind, but would very likely also lead to practical improvements in intelligent machine systems technology for manufacturing, construction, transportation, communication, health care, environmental restoration, waste management, security, and military systems. Such developments would have significant economic, social, and political benefits. But that is a subject for another paper.

This publication was prepared by a United States Government employee as part of his official duties and is, therefore, a work of the U.S. Government and not subject to copyright.

References

Albus, J.S. (1991). "Outline for a Theory of Intelligence". *IEEE Transactions on Systems, Man and Cybernetics*, **21**(3): p. 473-509

Albus, J.S. (1996). "The NIST Real-time Control System (RCS): An Approach to Intelligent Systems Research". *Journal of Experimental and Theoretical Artificial Intelligence* (in press)

Brooks, R.A. (1994). "Coherent Behavior from Many Adaptive Processes". *Proceedings of the Third International Conference on Simulation of Adaptive Behavior*. p. 22-29

Dickmanns, E.D. (1992). "A General Dynamic Vision Architecture for UGV and UAV". *Journal of Applied Intelligence* **2** p. 251-270

Dickmanns, E.D. (1995). "Parallel Use of Differential and Integral Representations for Realizing Efficient Mobile Robots". *7th International Symposium on Robotics Research*, October 1995, Munich

Rosenbloom, P.S., J.E. Laird, and A. Newell (Eds.) (1993) *The SOAR Papers*. Vol. 1 and 2 MIT Press, Cambridge, Mass.

Kosslyn, S.M. (1994). *Image and Brain: The Resolution of the imagery Debate*. MIT Press, Cambridge, Mass.

Schneiderman, H. and M. Nashman (1994). "Visual Tracking for Autonomous Driving," *IEEE Transactions on Robotics and Automation*, **10** (6) p. 769-775

TEAM (1996). Technologies for Enabling Agile Manufacturing (TEAM) Application Programming Interfaces, Internent location http://isd.cme.nist.gov/info/team

---

[1] Certain commercial equipment, instruments, or materials are identified in this report in order to facilitate understanding. Such identification does not imply recommendation or endorsement by the National Institute of Standards and Technology, nor does it imply that the materials or equipment identified are necessarily the best available for the purpose.

PERCEPTION AND MOTOR CONTROL

Self-Taught Visually-Guided Pointing for a Humanoid Robot

Matthew Marjanović, Brian Scassellati, Matthew Williamson*
545 Technology Square
Room NE43-920
Cambridge, MA 02139
maddog@ai.mit.edu, scaz@ai.mit.edu, matt@ai.mit.edu

Abstract

The authors implemented a system which performs a fundamental visuomotor coordination task on the humanoid robot Cog. Cog's task was to saccade its pair of two degree-of-freedom eyes to foveate on a target, and then to maneuver its six degree-of-freedom compliant arm to point at that target. This task requires systems for learning to saccade to visual targets, generating smooth arm trajectories, locating the arm in the visual field, and learning the map between gaze direction and correct pointing configuration of the arm. All learning was self-supervised solely by visual feedback. The task was accomplished by many parallel processes running on a seven processor, extensible architecture, MIMD computer.

1 Introduction

This paper is one of a series of developmental snapshots from the Cog Project at the MIT Artificial Intelligence Laboratory. Cog is a humanoid robot designed to explore a wide variety of problems in artificial intelligence and cognitive science (Brooks & Stein 1994). To date our hardware systems include a ten degree-of-freedom upper-torso robot, a multiprocessor MIMD computer, a video capture/display system, a six degree-of-freedom series-elastic actuated arm, and a host of programming language and support tools (Brooks 1996, Brooks, Bryson, Marjanovic, Stein, & Wessler 1996). This paper focuses on a behavioral system that learns to coordinate visual information with motor commands in order to learn to point the arm toward a visual target. Related work on Cog is also being presented at this conference, see (Ferrell 1996, Williamson 1996). Additional information on the project background can be found in (Brooks & Stein 1994, Irie 1995, Marjanovic 1995, Matsuoka 1995, Pratt & Williamson 1995, Scassellati 1995).

Given the location of an interesting visual stimulus in the image plane, the task is to move the eyes to foveate on that stimulus and then move the arm to point to that visual location. We chose this task for four reasons: First, the task is a fundamental component of more complex tasks, such as grasping an object, shaking hands, or playing "hide-and-seek" with small toys. Second, reaching to a visually stimulating object is a skill that children develop at a very early age (before the 5th month), and the development of this skill is itself an active area of research (Diamond 1990). Third, the task specification can be reformulated as a variety of behavioral responses. The task can be viewed as a pointing behavior (to show the location of a desired object), a reaching behavior (to move the arm to a position where the hand can begin to grasp an object), a protective reflex (to move the arm to intercept a dangerous object), or even as an occlusion task (to move the arm to block out bright lights or to hide an object from sight like the children's game "peek-a-boo"). Finally, the task requires integration at multiple levels in our robotic system.

To achieve visually-guided pointing, we construct a system that first learns the mapping from camera image coordinates $\vec{x} = (x, y)$ to the head-centered coordinates of the eye motors $\vec{e} = (pan, tilt)$ and then to the coordinates of the arm motors $\vec{\alpha} = (\alpha_0 ... \alpha_5)$. An image correlation algorithm constructs

*The authors receive support from a National Science Foundation Graduate Fellowship, a National Defense Science and Engineering Graduate Fellowship, and JPL Contract # 959333, respectively. Support for the Cog project is provided by an ONR/ARPA Vision MURI Grant (No. N00014-95-1-0600), a National Science Foundation Young Investigator Award (No. IRI-9357761) to Professor Lynn Andrea Stein, and by the J.H. and E.V. Wade Fund. Any opinions, findings, conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of our sponsors.

a saccade map $\vec{S} : \vec{x} \rightarrow \vec{e}$, which relates positions in the camera image with the motor commands necessary to foveate the eye at that location. Our task then becomes to learn the ballistic movement mapping from head-centered coordinates $\vec{e}$ to arm-centered coordinates $\vec{\alpha}$. To simplify the dimensionality problems involved in controlling a six degree-of-freedom arm, arm positions are specified as a linear combination of basis posture primitives. The ballistic mapping $\vec{B} : \vec{e} \rightarrow \vec{\alpha}$ is constructed by an on-line learning algorithm that compares motor command signals with visual motion feedback clues to localize the arm in visual space.

The next section describes the hardware of Cog's visual system, the physical design of the arm, and the computational capabilities of Cog. Section 3 gives a functional overview of the parallel processes that cooperate to achieve the pointing task. Section 4 describes details of the visual system: how the saccade map is learned and how the end of the arm is located in the visual field. Section 5 details the decomposition of arm movements into a set of linearly separable basic postures, and the learning algorithms for the ballistic map are explained in Section 6. Preliminary results of this learning algorithm and continuing research efforts can be found in Section 7.

2 Robot Platform

This section gives a brief specification of the physical subsystems of Cog (see Figures 1 and 2) that are directly relevant to our pointing task. We will describe the visual inputs that are available, the design and physical characteristics of the arm, and the processing capabilities of Cog's "brain". We have compressed much detail on the Cog architecture into this section for those readers interested in observing the progress of the project as a whole. Readers interested only in the pointing task presented here may omit many of these details.

2.1 Camera System

To approximate human eye movements, the camera system has four degrees-of-freedom consisting of two active "eyes" (Ballard 1989). Each eye can rotate about a vertical axis (pan) and a horizontal axis (tilt). Each eye consists of two black and white CCD cameras, one with a wide peripheral field of view ($88.6°(V) \times 115.8°(H)$) and the other with a narrow foveal view ($18.4°(V) \times 24.4°(H)$). Our initial experiments with the pointing task have used only the wide-angle cameras.

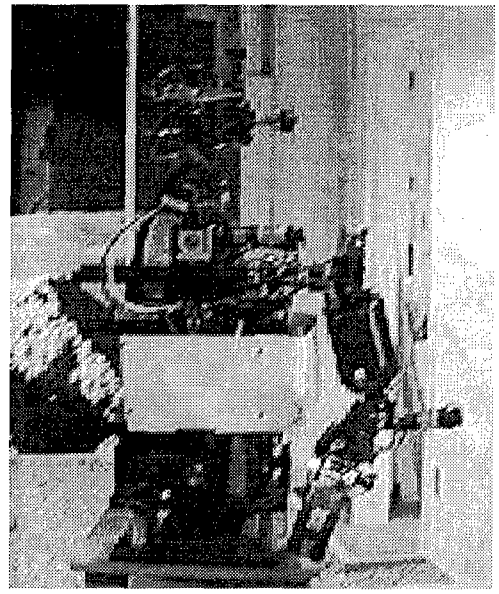


Figure 1: Cog, an upper-torso humanoid robot. Cog has two degrees-of-freedom in the waist, one in the shoulder, three in the neck, six on the arm, and two for each eye.

The analog NTSC output from each camera is digitized by a custom frame grabber designed by one of the authors. The frame grabbers subsample and filter the camera signals to produce 120×120 images in 8-bit grayscale, which are written at a frame rate of 30 frames per second to up to six dual-ported RAM (DPRAM) connections. Each DPRAM connection can be linked to a processor node or to a custom video display board. The video display board reads images simultaneously from three DPRAM slots and produces standard NTSC output, which can then be routed to one of twenty video displays.

2.2 Arm Design

The arm is loosely based on the dimensions of a human arm, and is illustrated in Figure 1. It has 6 degrees-of-freedom, each powered by a DC electric motor through a series spring (a series elastic actuator, see (Pratt & Williamson 1995)). The spring provides accurate torque feedback at each joint, and protects the motor gearbox from shock loads. A low gain position control loop is implemented so that each joint acts as if it were a virtual spring with variable stiffness, damping and equilibrium position. These spring parameters can be changed, both to move the arm and to alter its dynamic behavior. Motion of the arm is achieved by changing the equilibrium positions of the joints, not by commanding the

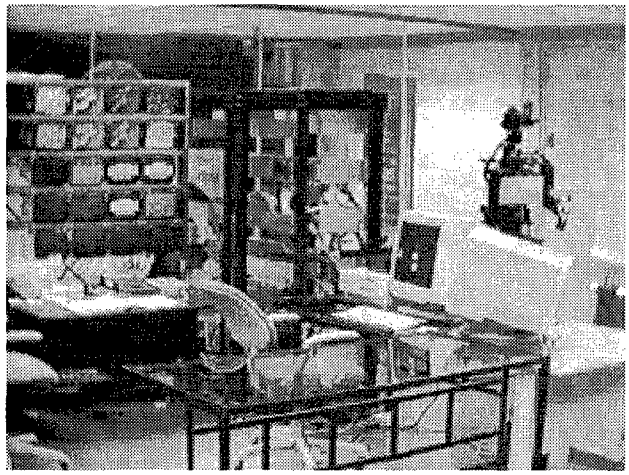


Figure 2: Supporting structure for Cog. The "brain" of the robot is a MIMD computer which occupies the racks in the center of this image. Video from the cameras or from the brain is displayed on a bank of twenty displays shown on the left. User interface and file storage are provided by a Macintosh Quadra. Cog itself is on the far right.

joint angles directly. There is considerable biological evidence for this spring-like property of arms (Zajac 1989, Cannon & Zahalak 1982, MacKay, Crammond, Kwan & Murphy 1986).

The spring-like property gives the arm a sensible "natural" behavior: if it is disturbed, or hits an obstacle, the arm simply deflects out of the way. The disturbance is absorbed by the compliant characteristics of the system, and needs no explicit sensing or computation. The system also has a low frequency characteristic (large masses and soft springs) which allows for smooth arm motion at a slower command rate. This allows more time for computation, and makes possible the use of control systems with substantial delay (a condition akin to biological systems). The spring-like behavior also guarantees a stable system if the joint set-points are fed-forward to the arm.

2.3 Computational System

The computational control for Cog is split into two levels: an on-board local motor controller for each motor, and a scalable MIMD computer that serves as Cog's "brain." This division of labor allows for an extensible and modular computer while still providing for rapid, local motor control.

Each motor has its own dedicated local motor controller, a special purpose board with a Motorola 6811HC11E2 microcontroller, which reads the encoder, performs servo calculations, and drives the motor with a 32KHz pulse-width modulated signal. For the eyes, the microcontroller implements a PID control law for position and velocity control, which is optimized for saccadic movement. For the arms, the microcontroller generates a virtual spring behavior at 1kHz. Similar motor control boards, with device-specific control programs, are used for body and neck motors.

Cog's "brain" is a scalable MIMD computer consisting of up to 239 processor nodes (although only eight are in use so far). During operation, the brain is a fixed topology network. However, the topology can be changed and scaled by adding additional nodes and connections. All components of the processing system communicate through 8K by 16 bit DPRAM connections, so altering the topology is relatively simple. Each node uses a standard Motorola serial peripheral interface (SPI) to communicate sensory information and control loop parameters with up to eight motor control boards at 50Hz. Each processor node contains its own 16MHz Motorola 68332 microprocessor mounted on a custom-built carrier board that provides support for the SPI communications and eight DPRAM connections. A Macintosh Quadra is used as the front-end processor for the user interface and file service (but *not* for any computation). Communication between the Quadra and the nodes of the MIMD computer is handled by a custom-built packet multiplexer box.

Each processor runs its own image of L, a compact, downwardly compatible version of Common Lisp that supports multi-tasking and multi-processing (Brooks 1996); and each uses IPS, a front end to L that supports communication between multiple processes (Brooks et al. 1996).

3 Task Overview

Figure 3 shows a schematic representation of the system architecture, at the process and processor level. The system can be decomposed into three major pieces, each developed semi-independently: visual, arm motor, and a ballistic map. The visual system is responsible for moving the eyes, detecting motion, and finding the end of the arm. The arm motor system maintains the variable-compliance arm and generates smooth trajectories between endpoints specified in a space of basis arm postures. The ballistic mapping system learns a feed-forward map from gaze position to arm position and generates reaching commands. Each of these subsystems is described in greater detail below.

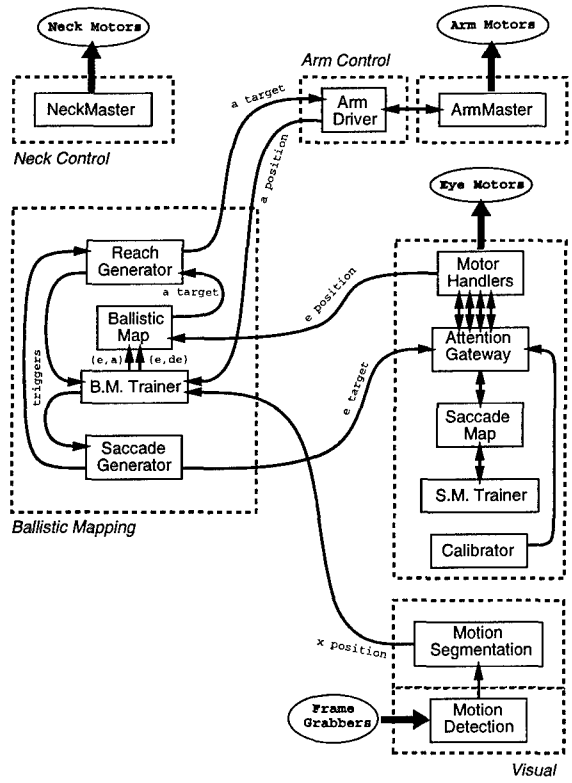


Figure 3: Schematic representation of the system architecture. Solid boxes are processes, dashed boxes indicate processor nodes. Messages pass between processors via dual-ported RAM connections. Image coordinates are represented by $\vec{x}$ positions, head-centered coordinates are represented by pan and tilt encoder readings $\vec{e}$, and arm positions are represented as linear combinations of the basis postures $\vec{\alpha}$.

For this first large-scale integration task implemented on Cog, we strove to meet a number of constraints, some self-imposed and some imposed by the hardware capabilities. The software architecture had to be distributed at both the processor and the process level. No single processor node had enough power to handle all the computation, nor enough peripheral control ports to handle the eleven motors involved. Within each processor, the system was implemented as collections of functionally independent but interacting processes. In the future we hope to implement more refined and elaborate behaviors by adding new processes to the existing network.

Although the basic activity for this particular task is sequential — foveate, reach, train, repeat — there is no centralized scheduler process. Rather, the action is driven by a set of triggers passed from one process to another. This is not a very important design consideration with the single task in mind; however as we add more processes, which act in parallel and compete for motor and sensor resources, a distributed system of activation and arbitration will become a necessity.

4 Visual System

The components of the visual system used in this task can be grouped into four functional units: basic eye-motor control, a saccade map trainer, a motion detection module, and a motion segmentation module. The eye-motor control processes maintain communication with the local motor control boards, initiate calibration routines, and arbitrate between requests for eye movement. The saccade trainer incrementally learns the mapping between the location of salient stimuli in the visual image with the eye motor commands necessary to foveate on that object. The motion detection system uses local area differences between successive camera images to identify areas where motion has occured. The output from the motion detection system is then grouped, segmented, and rated to determine the largest contiguous moving object. This segmented output is then combined with arm motor feedback by the ballistic map trainer (see Section 6) to locate the endpoint of the moving arm.

4.1 Eye Motor Control

The basic eye-motor control software is organized into a two-layer structure. In the lower layer, there is one process, called a handler, which maintains a continuous communication between the processor node and the local motor control board. In the upper layer is a single attentional gateway process which ensures that only one external process has control over the eyes at any given time. Currently, as soon as calibration has finished, the attentional gateway cedes control of the eye-motors to the ballistic map trainer. As more procedures begin to rely on eye movement, the attentional gateway will arbitrate between requests. Similar structures are used for the neck and arm motors, but do not appear in the Figure 3.

4.2 Learning the Saccade Map

In order to use visual information as an error signal for arm movements, it is necessary to learn the mapping between coordinates in the image plane and coordinates based on the body position of the robot. With the neck in a fixed position, this task simplifies to learning the mapping between image coordinates and the pan/tilt encoder coordinates of the eye motors. The behavioral correlate of this simplified task is to learn the pan and tilt positions necessary to saccade to a visual target. Initial experimentation revealed that for the wide-angle cameras, this saccade map is linear near the image center but rapidly diverged near the edges. An on-line learning algorithm was implemented to incrementally update an initial estimate of the saccade map by comparing image correlations in a local field. This learning process, the saccade map trainer, optimized a look-up table that contained the pan and tilt encoder offsets needed to saccade to a given image coordinate.

Saccade map training began with a linear estimate based on the range of the encoder limits (determined during calibration). For each learning trial, the saccade map trainer generated a random visual target location (x_t, y_t) and recorded the normalized image intensities $\bar{I}_t$ in a 16×16 patch around that point. The process then issued a saccade motor command using the current map entries. After the saccade, a new image $\bar{I}_n$ is acquired. The normalized 16×16 center of the new image is then correlated against the target image. Thus, for offsets x_0 and y_0, we sought to maximize the dot-product of the image vectors:

$$\max_{x_0,y_0} \left(\sum_i \sum_j \bar{I}_t(i,j) \cdot \bar{I}_n(x_0+i, y_0+j) \right) \quad (1)$$

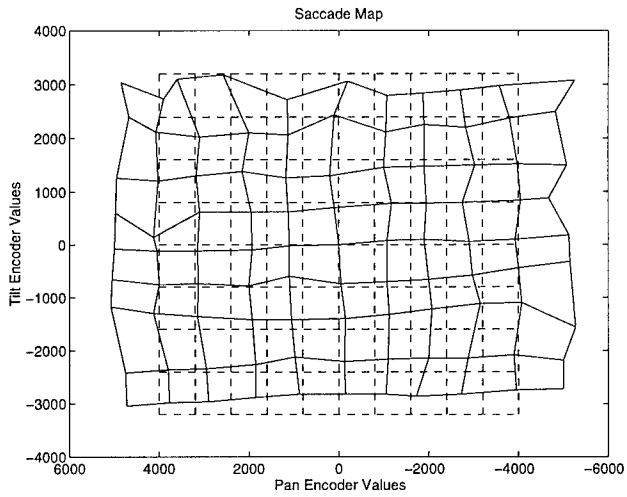


Figure 4: Saccade Map after 0 (dashed lines) and 2000 (solid lines) learning trials. The figure shows the pan and tilt encoder values for every tenth position in the image array within the ranges x=[10,110] (pan) and y=[20,100] (tilt).

Since each image was normalized, maximizing the dot product of the image vectors is identical to minimizing the angle between the two vectors. This normalization also gives the algorithm a better resistance to changes in background luminance as the camera moves. In our experiments, the offsets x_0 and y_0 had a range of $[-2, 2]$. The offset pair that maximized the expression in Equation 1, scaled by a constant factor, was used as the error vector for training the saccade map.

Note that a single learning step of this hill-climbing algorithm does not find the optimal correlations across the entire image. The limited search radius vastly increases the speed of each learning trial at the expense of producing difficulties with local maxima. However, in the laboratory space that makes up Cog's visual world, there are many large objects that are constant over relatively large pixel areas. The hill-climbing algorithm effectively exploited this property of the environment to avoid local maxima.

To simplify the learning process, we initially trained the map with random visual positions (x_t, y_t) that were multiples of ten in the ranges $[10, 110]$ for x_t (the pan dimension) and $[20, 100]$ for y_t (tilt). By examining only a subset of the image points, we could quickly train a limited set of points which would bootstrap additional points. Examining image points closer to the periphery was also unnecessary since the field of view of the camera was greater

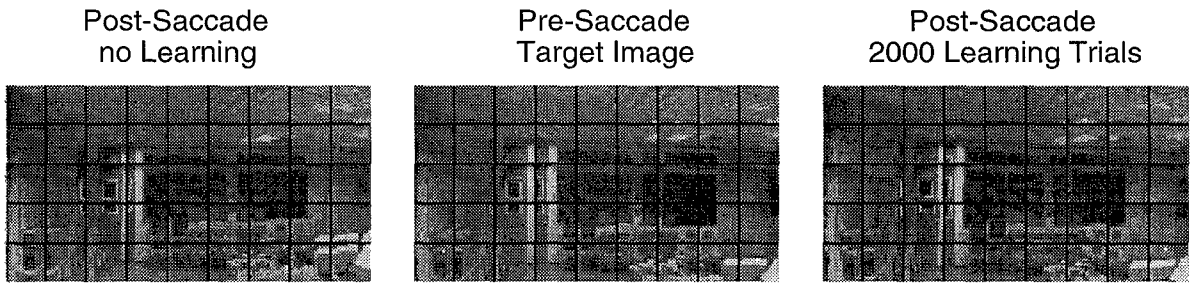


Figure 6: Expanded example of the visual learning of the saccade map. The center collage is the pre-saccade target images $\bar{I}_t$ for a subset of the entire saccade map. The left collage shows the post-saccade image centers with no learning. The right collage shows the post-saccade image centers after 2000 learning trials. The post-learning collage shows a much better match to the target than the pre-learning collage.

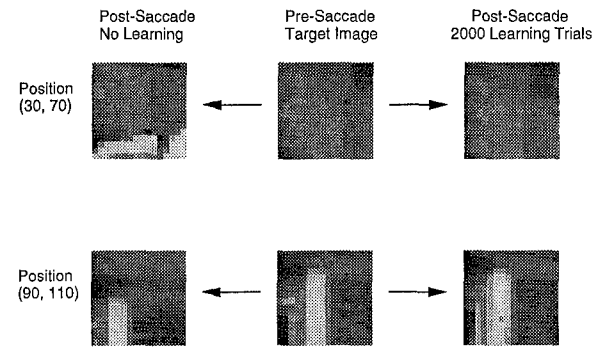


Figure 5: Two examples of the effects of the saccade map learning. The center set of images is the pre-saccade target image $\bar{I}_t$. The left image is the post-saccade image centers with no learning. The right image is the post-saccade image centers after 2000 learning trials. The post-learning images match the target more closely than the pre-learning images.

than the range of the motors; thus there were points on the edges of the image that could be seen but could not be foveated regardless of the current eye position. Figure 4 shows the data points in their initial linear approximation (dashed lines) and the resulting map after 2000 learning trials (solid lines). The saccade map after 2000 trials clearly indicates a slight counter-clockwise rotation of the mounting of the camera, which was verified by examination of the hardware. The training quickly reached a level of 1 pixel-error or less per trial within 2000 trials (approximately 20 trials per image location). Perhaps as a result of lens distortion effects, this error level remained constant regardless of continued learning.

Two examples of the visual effect of the learning procedure are shown in Figure 5. The center two images are the expected target images $\bar{I}_t$ recorded before the saccade for the image positions (30,70) and (90,110). Using the initial linear approximation with no learning, the post-saccade image $\bar{I}_n$ (shown at left) does not provide a good match to the target image (center). After 2000 learning trials, the difference in results is dramatic; the post-saccade image (shown to the right of the target) closely matches the pre-saccade target image. If the mapping had learned exactly the correct function, we would expect the pre-saccade and post-saccade images to be identical (modulo lens distortion). Visual comparison of the target images before saccade and the new images after saccade showed good match for all training image locations after 2000 trials. A larger set of examples from the collected data is shown in Figure 6.

4.3 Motion Detection and Segmentation

The motion detection and motion segmentation systems are used to provide visual feedback to the ballistic map trainer by locating the endpoint of the moving arm. The motion detection module computes the difference between consecutive wide-angle images within a local field. The motion segmenter then uses a region-growing technique to identify contiguous blocks of motion within the difference image. The bounding box of the largest motion block is then passed to the ballistic map trainer as a visual feedback signal for the location of the moving arm. In order to operate at speeds close to frame rate, the motion detection and segmentation routines were divided between two processors.

The motion detection process receives a digitized 120×120 image from the left wide-angle camera. Incoming images are stored in a ring of three frame buffers; one buffer holds the current image I_0, one buffer holds the previous image I_1, and a third buffer receives new input. The absolute value of the dif-

ference between the grayscale values in each image is thresholded to provide a raw motion image ($I_{raw} = T(|I_0 - I_1|)$). The raw motion image is then used to produce a motion receptive field map, a 40 × 40 array in which each cell corresponds to the number of cells in a 3 × 3 receptive field of the raw motion image that are above threshold. This reduction in size allows for greater noise tolerance and increased processing speed.

The motion segmentation module takes the receptive field map from the motion detection processor and produces a bounding box for the largest contiguous motion group. The process scans the receptive field map marking all locations which pass threshold with an identifying tag. Locations inherit tags from adjacent locations through a region grow-and-merge procedure. Once all locations above threshold have been tagged, the tag that has been assigned to the most locations is declared the "winner". The bounding box of the winning tag is computed and sent to the ballistic map trainer.

5 Arm Motion Control

5.1 *Postural Primitives*

The method used to control the arm takes inspiration from work on organization of movement in the spinal cord of frogs (Bizzi, Mussa-Ivaldi & Giszter 1991, Giszter, Mussa-Ivaldi & Bizzi 1993, Mussa-Ivaldi, Giszter & Bizzi 1994). These researchers electrically stimulated the spinal cord, and measured the forces at the foot, mapping out a force field in leg-motion space. They found that the force fields were convergent (the leg would move to fixed posture under the field's influence), and that there were only a small number of fields (4 in total). This lead to the suggestion that these postures were primitives that could be combined in different ways to generate movement (Mussa-Ivaldi & Giszter 1992). Details on the application of this research to robotic arms can be found in (Williamson 1996).

In Cog's arm the primitives are implemented as a set of equilibrium angles for each of the arm joints, as shown in Figure 7. Each primitive corresponds to a different posture of the arm. Four primitives are used: a rest position, and three on the extremes of the workspace in front of the robot. These are illustrated in Figure 8. Positions in space can be reached by interpolating between the primitives, giving a new set of equilibrium angles for the arm, and so a new end-point position. The interpolation is linear in primitive and joint space, but due to the non-linearity of the forward kinematics (end-point position in terms of joint angles), the motion in Cartesian space is not linear. However since only 4 primitives are used to move the 6 DOF arm, there is a large reduction in the dimensionality of the problem, with a consequent reduction in complexity.

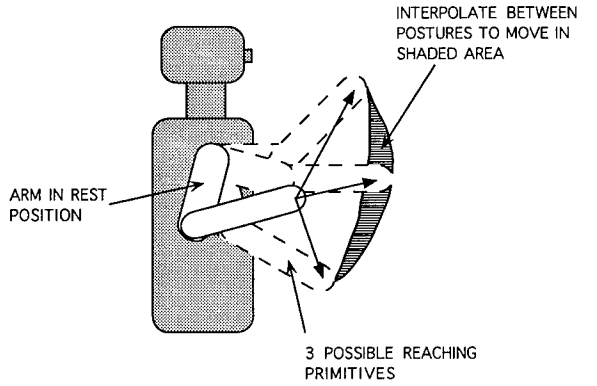


Figure 7: Primitives for the reaching task. There are four primitives: a rest position, and three in front of the robot. Linear interpolation is used to reach to points in the shaded area. See also Figure 8.

There are some other advantages to using this primitive scheme. There is a reduction in communication bandwidth as the commands to the arm need only set the rest positions of the springs, and do not deal with the torques directly. In addition the motion is bounded by the convex hull of the primitives, which is useful if there are known obstacles to avoid (like the body of the robot!).

5.2 *Reaching motion*

The reaching behavior takes inspiration from studies of child development (Diamond 1990). Children always begin a reach from a rest position in front of their bodies. If they miss the target, they return to the rest position and try again. This reaching sequence is implemented in Cog's arm. Infants also have strong grasping and withdrawal reflexes, which help them interact with their environment at a young age. These reflexes have also been implemented on Cog (Williamson 1996).

The actual motion takes inspiration from observations of the smooth nature of human arm motions (Flash & Hogan 1985). To produce a movement, the joints of the arm are moved using a smooth, minimum jerk profile (Nelson 1983).

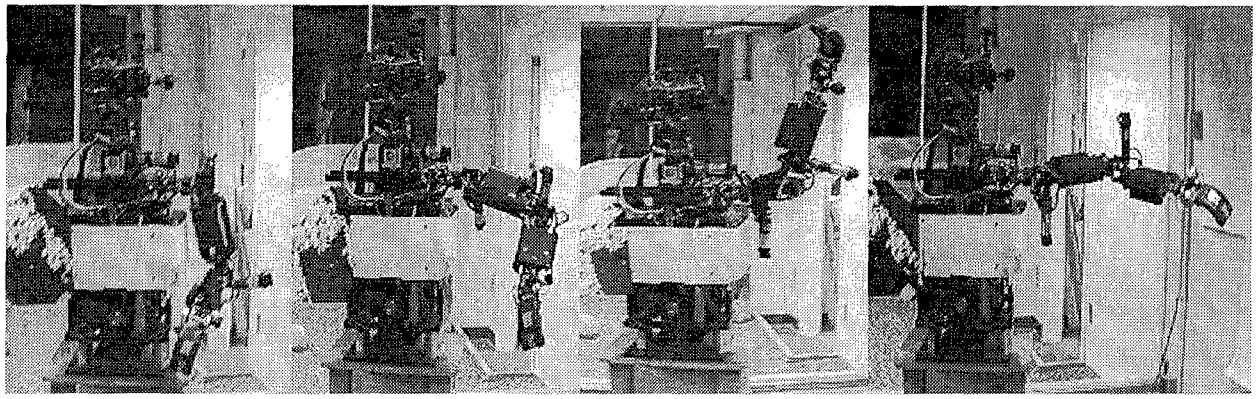


Figure 8: The basic arm postures. From left, "rest", "front", "up", and "side."

6 Ballistic Map

The ballistic map is a learned function $\vec{B}$ mapping eye position $\vec{e}$ into arm position $\vec{\alpha}$, such that the resulting arm configuration puts the end of the arm in the center of the visual field. Arm position is specified as a vector in a space of three basic 6-dimensional joint position vectors — the *reach* primitives (shown in Figure 8). There is also a fourth "rest" posture to which the arm returns between reaches.

The reach primitive coefficents are interpreted as percentages, and thus are required to sum to unity. This constrains the reach vectors to lie on a plane, and the arm endpoint to lie on a two-dimensional manifold. Thus, the ballistic map $\vec{B}$ is essentially a function $\mathcal{R}^2 \rightarrow \mathcal{R}^2$.

We attempted to select reach primitives such that the locus of arm endpoints was smooth and 1-to-1 when mapped onto the visual field. The kinematics of the arm and eye specify a function $\vec{E} : \vec{\alpha} \mapsto \vec{e}$ which maps primitive-specified arm positions into the eye positions which stare directly at the end of the arm. The ballistic map $\vec{B}$ is essentially the inverse of $\vec{E}$: we desire $\vec{E}(\vec{B}(\vec{e})) = \vec{e}$. If $\vec{E}$ is 1-to-1, then $\vec{B}$ is single-valued and we need not worry about learning discontinuous or multiple output ranges.

The learning techniques used here closely parallels the distal supervised learning approach (Jordan & Rumelhart 1992). We actually learned the forward map $\vec{E}$ as well as $\vec{B}$; this was necessitated by our training scheme. However, $\vec{E}$ is useful in that it gives an expectation of where to look to find the arm. This can be used to generate a window of attention to filter out distractions in the motion detection.

6.1 Map Implementation

The maps $\vec{B}$ and $\vec{E}$ are both implemented using a simple radial basis function approach. Each map consists of 64 Gaussian nodes distributed evenly over the input space. The nodes have identical variance, but are associated with different output vectors. The output of such a network ($\vec{y}$) for some input vector $\vec{i}$ is given by:

$$\vec{y} = \sum_k \vec{w}_k g_k(\vec{i}),$$

where

$$g_k(\vec{i}) = \exp(-\frac{1}{\sigma^2}\|\vec{i} - \vec{u}_k\|^2).$$

and $\vec{w}_k$ is a set of weights.

The ballistic map is initialized to point the arm to the center of the workspace for all gaze directions. The forward map is initialized to yield a centered gaze for all arm positions.

6.2 Learning the Ballistic Map

After the arm has reached out and its endpoint has been detected in the visual field, the ballistic map $\vec{B}$ is updated. However, since the error signal is a position in the image plane, the training cannot be done directly. We need to use the forward map $\vec{E}$ and the saccade map $\vec{S}$.

The current gaze direction $\vec{e}_0$ is fed through $\vec{B}$ to yield a reach vector $\vec{\beta}$ (β-space is a two dimensional parameterization of the α reach-primitive space). This $\vec{\beta}$ is sent to the arm to generate a reaching motion. It is also fed through the forward map $\vec{E}$ to generate an estimate $\vec{e}_p$ of where the arm will be in gaze-space after the reach. In an ideal world, $\vec{e}_p$ would equal $\vec{e}_0$.

After the arm has reached out, the motion detection determines the position $\vec{x}$ of the arm in pixel coordinates. If the reach were perfect, this would be the center of the image. Using the saccade map $\vec{S}$, we can map the difference in image (pixel) offsets between the end of the arm and the image center into gaze (eye position) offsets. So, we can use $\vec{S}$ to convert the visual position of the arm $\vec{x}$ into a gaze direction error $\Delta\vec{e}$.

We still cannot train $\vec{B}$ directly, since we have an e-space error but a β-space output. However, we can backpropagate $\Delta\vec{e}$ through the forward map $\vec{E}$ to yield a useful error term.

After all is said and done, we are performing basic least-mean-squares (LMS) gradient descent learning on the gaze error $\Delta\vec{e}$. For $\vec{B}$ defined by:

$$\vec{\beta} = \vec{B}(\vec{e}) = \sum_k \vec{w}_k g_k(\vec{e})$$

the update rule for the weights $\vec{w}_k$ is:

$$\Delta w_{ik} = -\eta \left(\Delta\vec{e} \cdot \frac{\partial \vec{F}}{\partial \beta_i} \right) g_k(\vec{e}).$$

for some learning rate η.

The forward map $\vec{F}$ is learned simultaneously with the ballistic map. Since $\vec{e} = \vec{e}_0 + \Delta\vec{e}$ is the gaze position of the arm after the reach, and $\vec{e}_p$ is the position predicted by $\vec{F}$, $\vec{F}$ can be trained directly via gradient descent using the error $(\vec{e}_p - \vec{e})$.

7 Results, Future Work, and Conclusions

At the immediate time of this writing, the complete system has been implemented and debugged, but has not been operational long enough to fully train the ballistic map. Initial results on small subsets of the visual input space show promising results. However, it will take some more extended training sessions before Cog has fully explored the space of reaches.

In addition to completing Cog's basic ballistic pointing training, our plans for upcoming endeavors include:

- incorporating additional degrees of freedom, such as neck and shoulder motion, into the model
- refining the arm finding process to track the arm during reaching
- expanding the number of primitive arm postures to cover a full three-dimensional workspace
- extracting depth information from camera vergence and stereopsis, and using that to implement reaching to and touching of objects.
- adding reflexive motions such as arm withdrawal and a looming response, including raising the arm to protect eyes and head
- making better use of the inverse ballistic map in reducing the amount of computation necessary to visually locate the arm.

This pointing task, albeit simple when viewed alongside the myriad complex motor skills of humans, is a milestone for Cog. This is the first task implemented on Cog which integrates major sensory and motor systems using a cohesive distributed network of processes on multiple processors. To the authors, this is a long-awaited proof of concept for the hardware and software which have been under development for the past two and a half years. Hopefully, this task will be a continuing part of the effort towards an artificial machine capable of human-like interaction with the world.

8 Acknowledgments

The authors wish to thank the members of the Cog group (past and present) for their continual support: Mike Binnard, Rod Brooks, Cynthia Ferrell, Robert Irie, Yoky Matsuoka, Nick Shectman, and Lynn Stein.

References

Ballard, D. (1989), 'Behavioral Constraints on Animate Vision', *Image and Vision Computing* **7:1**, 3–9.

Bizzi, E., Mussa-Ivaldi, F. A. & Giszter, S. F. (1991), 'Computations underlying the execution of movement: A biological perspective', *Science* **253**, 287–291.

Brooks, R. (1996), L, Technical report, IS Robotics Internal Document.

Brooks, R. & Stein, L. A. (1994), 'Building Brains for Bodies', *Autonomous Robots* **1:1**, 7–25.

Brooks, R., Bryson, J., Marjanovic, M., Stein, L. A., & Wessler, M. (1996), Humanoid Software, Technical report, MIT Artificial Intelligence Lab Internal Document.

Cannon, S. & Zahalak, G. I. (1982), 'The mechanical behavior of active human skeletal muscle

in small oscillations', *Journal of Biomechanics* **15**, 111–121.

Diamond, A. (1990), *Development and Neural Bases of Higher Cognitive Functions*, Vol. 608, New York Academy of Sciences, chapter Developmental Time Course in Human Infants and Infant Monkeys, and the Neural Bases, of Inhibitory Control in Reaching, pp. 637–676.

Ferrell, C. (1996), Orientation Behavior Using Registered Topographic Maps, Society of Adaptive Behavior. In these proceedings.

Flash, T. & Hogan, N. (1985), 'The Coordination of Arm Movements:An Experimentally Confirmed Mathematical Model', *Journal of Neuroscience* **5**(7), 1688–1703.

Giszter, S. F., Mussa-Ivaldi, F. A. & Bizzi, E. (1993), 'Convergent Force Fields Organized in the Frog's Spinal Cord', *Journal of Neuroscience* **13**(2), 467–491.

Irie, R. (1995), Robust Sound Localization: An Application of an Auditory Perception System for a Humanoid Robot, Master's thesis, MIT Department of Electrical Engineering and Computer Science.

Jordan, M. I. & Rumelhart, D. E. (1992), 'Forward Models: supervised learning with a distal teacher', *Cognitive Science* **16**, 307–354.

MacKay, W. A., Crammond, D. J., Kwan, H. C. & Murphy, J. T. (1986), 'Measurements of human forearm posture viscoelasticity', *Journal of Biomechanics* **19**, 231–238.

Marjanovic, M. (1995), Learning Functional Maps Between Sensorimotor Systems on a Humanoid Robot, Master's thesis, MIT Department of Electrical Engineering and Computer Science.

Matsuoka, Y. (1995), Embodiment and Manipulation Learning Process for a Humanoid Hand, Master's thesis, MIT Department of Electrical Engineering and Computer Science.

Mussa-Ivaldi, F. A. & Giszter, S. F. (1992), 'Vector field approximation: a computational paradigm for motor control and learning', *Biological Cybernetics* **67**, 491–500.

Mussa-Ivaldi, F. A., Giszter, S. F. & Bizzi, E. (1994), 'Linear combinations of primitives in vertebrate motor control', *Proceedings of the National Academy of Sciences* **91**, 7534–7538.

Nelson, W. L. (1983), 'Physical Principles for Economies of Skilled Movements', *Biological Cybernetics* **46**, 135–147.

Pratt, G. A. & Williamson, M. M. (1995), Series Elastic Actuators, *in* 'Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS-95)', Vol. 1, Pittsburg, PA, pp. 399–406.

Scassellati, B. (1995), High Level Perceptual Contours from a Variety of Low Level Features, Master's thesis, MIT Department of Electrical Engineering and Computer Science.

Williamson, M. M. (1996), Postural primitives: interactive behavior for a humanoid robot arm, Society of Adaptive Behavior. In these proceedings.

Zajac, F. E. (1989), 'Muscle and tendon:Properties, models, scaling, and application to biomechanics and motor control', *CRC Critical Reviews of Biomedical Engineering* **17**(4), 359–411.

A Self-Organizing Model of the Antennal Lobes

Rainer Malaka[1], Stefan Schmitz[1], and Wayne M. Getz[2]

[1] Institut für Logik, Komplexität und Deduktionssysteme, Universittät Karlsruhe, D-76128 Karlsruhe, Germany, malaka@ira.uka.de

[2] Department of ESPM, University of California at Berkeley, CA 94720-3112, USA, getz@nature.berkeley.edu

Abstract

For most animals their olfactory system provides vital interface with the outside world. Odors, however, are typically unstable signals, fluctuating in time, space and in concentration. Moreover generalist odor detection systems have to be able to process many odors, most of which are of incidental importance. In this paper, we present a self-organizing model of the antennal lobe of the insect brain. The learning rule used enables the model to produce stable coding of relevant odors, where the coding is invariant against temporal and spatial fluctuations. The model includes developmental learning as well as on-line adaptive tuning to a changing environment. Thus the model describes a system that would enable individuals to survive in an uncertain world of odors.

1 Introduction

For many species, their chemical senses are their most important interface with their surrounding environment. Among other things, chemical senses are critical to finding food, avoiding poisonous substances, finding mates, and identifying kin. Odor coding in animals is less well understood, however, than visual and auditory coding. Unlike visual and auditory stimuli, odors cannot be described in a low-dimensional stimulus space (wavelength, frequency, spatial dimensions, or amplitude). Each odor ligand forms a complex three dimensional electrical charge distribution that determines whether it can form a temporary association with receptor molecules on the surface of olfactory sensory cells. Some of these receptors bind to a range of ligand molecules while others, such as those involved with the detection of pheromones are highly specific.

In contrast to the highly specialized pheromone detection systems that are found, for example, in the males of a number of species of insects, generalist odor processing systems have to be able to encode several hundreds or thousands of odors. Humans, of course, can perform such tasks, as can insects, such as worker honey bees. Akers and Getz (Akers & Getz, 1992, Akers & Getz, 1993) have shown that honey bee olfactory receptors neurons (ORNs) cannot be cleanly organized into a few classes, as previously suggested by Vareschi (Vareschi, 1971). Most of the ORNs are broadly tuned and respond with a variety of response profiles, including strong synergistic and inhibitory effects, to mixtures of odors. Additionally, odors are noisy signals incorporating spatial and temporal fluctuations and a high level of background stimuli. On the basis of these constraints, several tasks might be solved by antennal lobes, the primary olfactory neuropile:

- The high dimension of the input signal provided by 30000 ORNs must be reduced.
- The internal odor coding should provide stable odor images that enable recognition of an odor independently of small concentration fluctuations.
- Due to spatial variations of odors, the antennal stimulation needs not to be the same for two presentations of the same odor. This must be counterbalanced by neural processing in order to generate an invariant coding.
- In a natural context, many olfactory stimuli occur in a background that must be separated from the interesting signals.
- Animals, such as honey bees can live in an array of ecological systems, each of which has its own unique set of odor cues associated with suitable sources of food. Thus the generalist olfactory system of the honey bee should be optimized for the set of environmental olfactory cues in which the animal lives.
- Since the environment does not stay the same during the life time of a honey bee, e.g., some flowers have nectar only for a short period of time, the animal has to adapt to new odors, as well as encoding relevant odors more reliably than extraneous stimuli.

The insect antennal lobes have many features in common with the vertebrate olfactory bulb (Boeckh *et al.*, 1990): the glomerular structure, the convergence of a large number of ORNs onto a low number of glomeruli, inhibitory feedback mediated by local interneurons, and a low number of projection neurons per glomerulus that

provide antennal lobe output (Halasz, 1989). As for the ORNs, there seems to be also no distinct specialization on the glomerular or projection neuron level (Boeckh *et al.*, 1990).

The antennal lobes are divided into glomeruli, which are spheroidal regions housing synapses between ORNs, local interneurons and projection neurons (Tolbert & Hildebrand, 1981, Homberg *et al.*, 1989). There are ten to hundreds of glomeruli in various insect species. Their number and arrangement is species specific (Rospars, 1988). Projection neurons within a glomerulus have both input and output synapses (Malun, 1991).

The morphology of the interneurons and the projection neurons clearly describes a recurrent network involving projection neurons and local feedback through arborizations of local interneurons.

Structure and connectivity of antennal lobe neurons underlie changes that are influenced by external processes and neural activity. Three different kinds of plasticity can be observed in the antennal lobes: developmental formation of the antennal lobe structure, non-associative plasticity such as habituation and sensitization, and associative changes in synaptic strengths.

During pupal development, ORNs elaborate dendrites entering the developing sensilla and send axons to the antennal lobes, where they connect to antennal lobe neurons (Masson & Mustaparta, 1990). First glomerular-like structures appear on the third pupal day. On the seventh day, the antennal lobe seems to be fully established and no significant differences to the adult animal remain. Masson and Mustaparta (Masson & Mustaparta, 1990) hypothesize that peripheral activity might trigger the development of the antennal lobe. This hypothesis is confirmed by two observations. First, if the antennal input is removed, the evolved antennal lobe structure is much smaller, only a few glomerular-like structures are built (Rybak & Eichmüller, 1993). Second, if the antennal input is grafted to male sensory neurons, as was done in the female moth *Manduca sexta*, the antennal gynandromorphs formed antennal lobes containing certain cells that are morphologically and physiologically male-like. Even the behavior of these transsexual animals shows typical male components, such as pheromonal attraction (Schneidermann *et al.*, 1982). Thus antennal input seems to be crucial in early development of the antennal lobe.

Non-associative plasticity covers changes in the response to a stimulus that are not dependent on a stimulus-specific pre-treatment. Often they do not last very long and can be reverted. Habituation and sensitization are the most prominent non-associative behavioral changes. Besides this evidence for non-associative learning, some experimental results indicate that associative learning also takes place in the antennal lobes. Since the VUMmx1 neuron, as identified in the honey bee, is know to represent the unconditioned stimulus (US) during classical conditioning, and this neuron also projects into the antennal lobes which process the conditioned stimulus (CS), this neuron is a putative convergence site of the CS and US pathways (Hammer, 1993, Hammer & Menzel, 1995). An experiment that substitute the VUMmx1 activation with local injections of its transmitter octopamine demonstrates that injections to the antennal lobes can substitute for the sucrose US. This experiment thus indicate that the antennal lobes are involved in associative learning (Hammer, 1995).

The vertebrate olfactory bulb on the other hand also exhibits associative plasticity. In classical conditioning experiments with neonate rats, odor response 2-DG images of the bulb significantly changed to a more focused activity after training. Interestingly, this effect occurred with both appetitive and aversive stimuli, and also to odors of females to whom they had previously been exposed (Wilson & Sullivan, 1994). Thus associative learning of *important* odors might occur, where the neural coding itself might not necessarily indicate the *meaning* of the odor.

Several models have been proposed for the vertebrate olfactory bulb and for the olfactory neuropile of the terrestrial mollusk, *Limax maximus*. This limax work was pioneered by Hopfield, Gelperin, Tank and is basically a simple Hopfield model that classifies odors by projecting the odors into the attractors of the model (Gelperin *et al.*, 1986, Gelperin & Hopfield, 1988, Hopfield, 1988, Gelperin *et al.*, 1989). Some models have been proposed that simulate the oscillatory and chaotic attractor modes in the olfactory bulb (Freeman, 1987, Li & Hopfield, 1989, Eisenberg *et al.*, 1989, Yao & Freeman, 1990). Schild and Riedel outlined a model for the olfactory bulb with linear transformations from ORNs to glomeruli and onto mitral cells and lateral inhibitory mechanisms, where odor detection is mainly achieved by the feed-forward transformations. The recurrent mechanisms act as high-pass filter to sharpen the information processed by the glomeruli (Schild, 1988, Schild & Riedel, 1989, Shepherd, 1992). The model, however, does not include feedback that are capable of simulating dynamic odor responses. To model plasticity, a Hebbian learning rule has been included (Schild & Riedel, 1989).

Getz (Getz, 1991) proposed a model based on Hopfield networks for olfaction in general, but few models specifically of insect olfactory systems exist. Some modeling approaches have been proposed that simulate the dynamical spatio-temporal across-fiber patterns observed in the neural activity of the antennal lobes in specialized pheromone detecting systems (Linster *et al.*, 1993, Linster *et al.*, 1994) and in generalist systems (Malaka, 1995, Malaka & Hammer, 1996, Linster & Masson, 1996).

Only a few models focus onto learning rules. Most models only incorporate simple *one-shot* Hebbian learning rules where all patterns are presented once to com-

pute all network-weights in an *off-line* learning scheme (e.g., (Eisenberg *et al.*, 1989, Yao & Freeman, 1990)). Hopfield introduced an olfactory bulb model with adaptable connections and a learning rule that has some similarities to Földiák's learning rule (Földiák, 1989). After learning, single neurons are capable of coding the concentrations of odors within a compound, and of simultaneously coding several odors that have independent fluctuations in concentration over time (Hopfield, 1991, Hendin *et al.*, 1994). Weaknesses are the non-adaptive synapses from ORNs to the network neurons and that the coding is terms of *grandmother cells*, that is, each odor is encoded by the activity of one cell.

In this paper, a self-organizing model is introduced that incorporates on-line learning rules. The model learns to encode odors irrespective of their spatial distribution on the antenna and thus provides a mechanism for encoding odors that whose concentration fluctuates both in space and time. Noisy input, i.e., small concentration differences, can be compensated by the network. The learning mechanism can be either thought of as self-organizing principle that operates during pupal development and is triggered by a genetic program that selectively activates ORN neurons, or as an on-line learning scheme that helps the animal to continuously adapt its coding to unpredictable changes in the surrounding world. It thus enables the animal to survive in uncertain environments.

2 A Self-organizing Model

Self-organization is a neural principle that classifies and reorganizes sensory input data in an unsupervised learning scheme. Self-organizing neural networks usually transform input signals to a code that may be better suitable for further processing in subsequent networks (Kohonen, 1988, Ritter *et al.*, 1991). The coding can result in a classification or a map of the input data, where only one representative neuron responds to a stimulus, or it can be distributed throughout the network of neurons.

Since no teaching signal is available in unsupervised learning, self-organizing algorithms have to find inherent structures and features in the input data. Two types of inherent features can be distinguished, spatial and temporal ones. If the data has a spatial structure, such as the two dimensional structure of image data, features can be extracted from local neighborhoods. If concurrent activation is meaningful, features can be extracted from the dynamics of the input.

For high dimensional ORN signals generated by the antenna, typically spatial patterns do not carry much information, since each ORN represents a single point (or small neighborhood) and thus no reasonable features can be extracted from local neighborhoods of ORNs. On the other hand, temporal information might give good

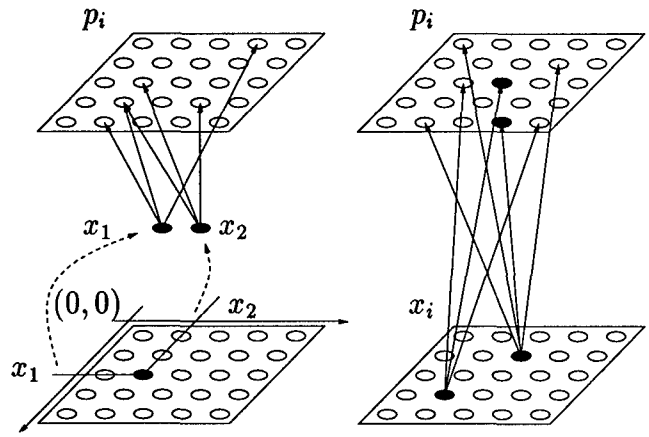


Figure 1: Architecture of the two principles of self-organization. Left: Kohonen's model uses the coordinates of active units as inputs. Right: von der Malsburg uses a full input matrix of sensory neurons

clues for structuring the ORN responses. Assuming that concurrent ORN activity is due to the same odor stimulation, these ORNs can be mapped onto the same odor representation.

Two popular models of unsupervised learning algorithms that self-organize are Kohonen's self-organizing feature maps and von der Malsburg's dynamic link architecture (Malsburg, 1973, Kohonen & Oja, 1976, Willchaw & Malsburg, 1976, Kohonen, 1982, Konen & Malsburg, 1992). Both models are designed to describe the retinotopic mapping of sensory input onto cortical layers while preserving the topology of the input data. Kohonen's self-organizing feature maps use a two-dimensional input signal that corresponds to the two coordinates of the input points. By contrast, the retinal information in von der Malsburg's models is a distributed coding, where each sensory neuron has its own activation (Fig.1). In Kohonen's models one input point is presented at one time step and spatial features of input points can easily be obtained by the distance between them. In the output layer, a winner-take-all algorithm selects the neuron with the highest response. A competitive learning rule is then applied to the winning neuron and with a smaller learning rate to its neighbors. Eventually, the given topology of the output layer maps the topology of the input data. Von der Malsburg uses a two-dimensional input array of retinal neurons that is spontaneously excited by so called *blob* activities (Konen & Malsburg, 1992). Thus, at one time step, sensory neurons in a local neighborhood are simultaneously active and the output layer can use this temporal information of concurrent activation to detect neighboring input lines and to build a retinotopic map of the input.

The low-dimensional sensory coding and the use of spatial features in Kohonen's network does not fit to the neural information processing in the antennal lobe.

The input is very high dimensional and spatial information is rather useless. Von der Malsburg's dynamic links use temporal information and map a distributed input coding onto a distributed output coding and thus can be used for modeling the antennal lobes. In the model description shown below, it will be shown that activity blobs can not only select neighboring cells, but also similar tuned cells.

In neuroscience, self-organizing models are typically investigated in the context of early development of the mapping of somatic, auditory and visual sensory input onto higher brain regions. This mapping is topological and it is not assumed that the wiring of all input fibers is genetically programmed (Martin, 1991). Besides the retinotopic mapping of visual signals, somatotopic mappings and tonototopic mappings of somatic and auditory signals have been modeled using self-organizing learning principles (Ritter & Schulten, 1986, Martinetz *et al.*, 1988). Self-organization, however, can also be a reorganization that may be active also in the adult. Ritter et al. investigated the cortical reconfiguration of monkeys after making lesions that removed parts of their sensory input and found that self-organizing feature maps are a good model for the resulting sensory map (Ritter & Schulten, 1986, Ritter *et al.*, 1991).

Here we introduce a self-organizing model of the antennal lobe. Similar to von der Malsburg's dynamic link architecture, it will use temporal information to map the sensory input onto a distributed output coding. This temporal information is produced by the temporal structure of odor plumes in the air. It can be shown that this new model learns to code odors in a stable way, independent of their spatial distribution on the antenna.

2.1 Odor Representation at the Sensory Input

Odor molecules are released from their source by evaporation and start moving randomly around in a Brownian motion which leads to diffusion. The odor concentration decreases and when the odor reaches the free wind, a still coherent and diffusing trail of odor molecules moves away from the source. When the odor plume reaches a specific size (roughly the Kolmogoroff scale), due to turbulence in the air generated by mechanical forces and buoyancy, turbulent diffusion starts and the plume breaks down into a cloud of unevenly distributed filaments (Murlis, 1986, Murlis *et al.*, 1992). Finally, odor packages with a steady concentration arise and are carried through the air. These stages of molecular transportation can, at different scales, be easily observed in smoke plumes from cigarette smoke or from chimneys, whereas the size of the plumes and the time scale of diffusion mainly depends on so called atmospheric eddies, which are large in higher atmospheric regions and small close to the ground (Murlis, 1986).

Thus, due to the physics of odor distribution, a specific odor with a specific concentration will never appear evenly distributed on the antenna. The insect will rather cross the fine odor filaments or receive odor packets. This leads to small odor puffs that cross the antenna in non-integrated intervals, even though the source delivers the odor steadily at a constant concentration. Moreover, the odor puffs migrate over the antenna and can reach the antenna at different locations.

These observations lead to the assumption that the odor image received by the antenna may look very different each time a specific odor is presented. The concentration distribution on the antenna may vary depending on the location where an odor puff hits the antenna. An additional problem might be the distinction between an odor mixture and two odors coming from different sources.

A good coding, however, should ensure that an odor is always represented in the same way independently of the spatio-temporal arrival processes of odor molecules on the antenna. Before we introduce a self-organizing model that produces an invariant odor coding, a simple model of odors and their spatio-temporal distribution is presented.

Odors are mixtures of elementary odorants. For a given set of k elementary odorants, an odor (or odor compound) can be described by a k-dimensional vector $O = (O_1, \ldots, O_k)$, where O_i is the concentration of the i-th elementary odor.

An odor O is now delivered in small odor packages, which cross the antenna. We assume the odor distribution within an odor package to be Gaussian. The antenna is modeled as a one-dimensional array of receptor neurons, ignoring the two-dimensionality of the surface of the antenna. We now denote the sites of the receptor neurons on the antenna with indices $1, \ldots, m$. Thus each receptor neuron has an index corresponding to its position on the antenna. The distance between each two neighboring receptor neurons is always the same.

The Gaussian concentration distribution of an odor package with center at the receptor neuron with index μ and variance σ, can now be described by

$$dist \quad : \quad \{1, \ldots, m\} \times \mathbb{R} \times \mathbb{R} \rightarrow [0, 1] \qquad (1)$$

$$dist \quad (i, \mu, \sigma) \quad = e^{-\frac{(i-\mu)^2}{2\sigma^2}} , \qquad (2)$$

which assigns every receptor neuron index i a value between 0 and 1, i.e., a concentration between zero and maximum. The antennal odor concentration of a given odor package is now modeled by a one-dimensional array

$$a \quad = \quad (a_1, \ldots, a_m) \qquad (3)$$

$$a_i \quad = \quad a_i(O, \mu, \sigma) = dist(i, \mu, \sigma) \cdot O \, . \qquad (4)$$

The antennal odor concentration a_i corresponds to the

actual odor input to the i-th receptor neuron on the antenna. Thus the antennal image of an odor on the antenna not only depends on the odor, but also on the shape of the odor package and the location where it hits the antenna. In the next step, the temporal aspect of odor package fluctuations has to be integrated. In our model we simply let the packages move around the antenna by shifting the mean μ over the antenna.

2.2 Model Formulation

The model is designed to investigate possible soft-wiring mechanisms that exist between olfactory receptor and antennal lobe neurons and among the antennal lobe neurons themselves. Here, we do not focus onto the dynamics of the antennal lobe output as we did in the last section. Rather, we formalize our idea of self-organization on the structure and topology of the basic model without interneurons and using a Hebbian type learning rule with symmetric weights (Ragg *et al.*, 1995). Thus only the most important neural components, receptor neurons and projection neurons, are modeled explicitly. The function of the local interneurons is hidden in the learning rule and the weight matrices. Even though the model is more abstract than those discussed above, the literature reviewed in the previous section suggests how to make the dynamic coding of the model more plausible; that is, by integrating interneurons with asymmetrical weights.

The learning rule introduced below is a Hebbian rule that forces the weights between the network neurons to be symmetric and allows us to evaluate the coding on the basis of stable states. Therefore, the normalized correlation $\overline{\text{corr}}$, i.e., the cosine of the angle between stable states are applied for comparison of the coding for two odors.

ORNs on the antenna form the input layer of the network. Each ORN is modeled either by a simulated ORN (Malaka *et al.*, 1995) or by simpler artificial models. Specifically, we randomly distributed l types of receptors onto the m receptor neuron slots assigned to the antenna using an index function

$$t : \{1, \ldots, m\} \rightarrow \{1, \ldots, l\} \tag{5}$$

which assigns the receptor type $t(i)$ to receptor neuron i. The activation of the receptor neurons x_i for a particular odor package on the antenna can now be computed using the effect functions r_i, $i = 1, \ldots, l$ of the receptor types and the antennal odor concentrations a_i

$$\begin{aligned} x_i(\mu, \sigma, O) &= r_{t(i)}(a_i(\mu, \sigma, O)) \\ &= r_{t(i)}(dist(i, \mu, \sigma) \cdot O) \end{aligned} \tag{6}$$

where O, μ and σ describe the odor package. A typical antennal odor response involving 200 antennal ORNs x_i and three ORN types r_j is given in Fig.2.

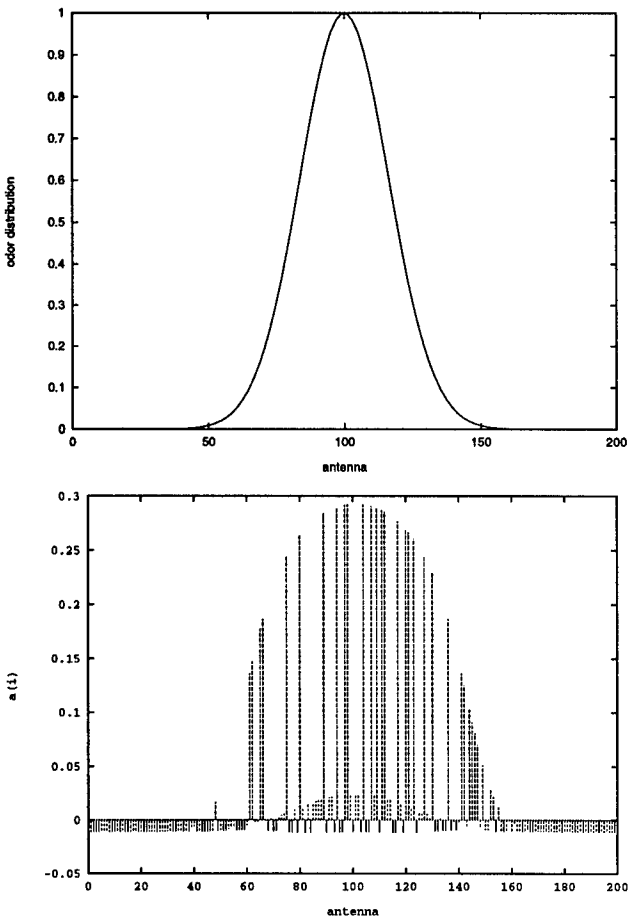


Figure 2: Odor distribution and antennal activation. Upper panel: Odor distribution $dist(i, \mu, \sigma)$ on the antenna with center in the middle of the antenna. Lower distribution: ORN response to the same distribution of a particular odor. Three ORN types are distributed on the antenna. The single ORN responses $x_i(\mu, \sigma, O)$ are given as solid, dotted and dashed bars for each type, respectively

The central layer of the network consists of a set of n projection neurons $p = (p_1, \ldots, p_n)$ that have recurrent connections and receive input from the receptor neuron layer. The neuronal model is a discrete time and continuous state model represented by the difference equations

$$\begin{aligned} p_i(t+1) &= f(rp_i(t) + \\ &\quad \Delta(\sum_{j=1}^{n} w_{ij} p_j(t) + \sum_{j=1}^{k} v_{ij} x_j(t) + \theta)) \end{aligned} \tag{7}$$

where $f = f_{sig}$ is a sigmoidal function, that restricts the output to values between 0 and 1. Projection neuron j is connected to neuron i with weight w_{ij}. The weight v_{ij} gives the synaptic strength of the connection of input x_j to projection neuron i, where v_{ij} is restricted to positive values, i.e., ORNs are always excitatory. The transmission time delay is set to the unit step Δt. The negative

constant bias θ is used to allow some inhibition, since $v_{ij} \geq 0$.

In our simplified model, the weights between projection neurons are symmetrical and thus with a constant input, the network state converges to a stable state, i.e. attractor. For each antennal activation and each initial activation the according stable state can be computed by relaxation of the network

$$p^{\infty} : 0, \ldots, m \times \mathbb{R} \times \mathbb{R}^k \times [0,1]^n \rightarrow [0,1]^n$$

$$p^{\infty}(\mu, \sigma, O, p^0) = \lim_{t \rightarrow \infty} p(t) \qquad (8)$$

for a given odor package defined by μ, σ, O and an initial state p^0. In practical computations $p(t)$ is iterated until the absolute change in each component $p_i(t)$ decreases a small constant. If, as we did in the simulations, the variances σ are set always to the same values, we simply write $p^{\infty}(\mu, O, p^0)$ instead of $p^{\infty}(\mu, \sigma, O, p^0)$.

The responses of the network to two odor packages μ_1, O^1 and μ_2, O^2 and a given variance σ can now be compared by the overlap ov of the stable states that emerge with the different network inputs

$$\begin{aligned} & ov(O_1, O_2, \mu_1, \mu_2) \\ & = \overline{\text{corr}}(p^{\infty}(\mu_1, O^1, p^0), p^{\infty}(\mu_1, O^2, p^0)) \end{aligned} \qquad (9)$$

for a given initial state p^0. The maximal overlap value $ov(O_1, O_2, \mu_1, \mu_2) = 1$ means that the activation vectors point in the same direction, i.e. they are similar, and an overlap $ov(O_1, O_2, \mu_1, \mu_2) = 0$ indicates that the activation vectors are orthogonal, i.e. dissimilar.

2.3 Learning Algorithm

The learning algorithm consists of two learning rules, one for the input weight matrix $V = (v_{ij})$, and one for the network weight matrix $W = (w_{ij})$. The goal of the learning process is to make use of the spatio-temporal odor distributions on the antenna and to generate a coding that is invariant against spatial fluctuations and to some extent against fluctuations in the odor concentrations.

During odor stimulation a population of neurons, overlapping with the odor package, receives sensory input, and those neurons sensitive to this odor are concurrently active. Due to the migration of the odor package on the antenna, at the next time step, another population of receptor neurons responds to that odor. In our simulations, the migration of an odor package is modeled by a linear shift of the Gaussian odor concentration over the antenna, which models the stimulation of the antenna by natural odor plumes. Another possibility of stimulus generation is a spontaneous genetically triggered antennal activation program that selectively activates ORNs that express similar receptor proteins. Such a process could take place during pupal development and would not need external input. This would also be an explanation for pupal ORN activity and is consistent with the theories from vertebrate development. The learning rule tries to link those neurons together that have the same selectivity. The capacitance incorporated in the projection neuron model enables the model to not only link concurrently active receptor neurons, but also to link consecutive active neurons together.

The resulting mapping of receptor neurons to projection neurons is not a topographic map of the antenna, but a topographic map of odors. Additional constraints added to the learning rule enforce the weights to have an antennal lobe-like connectivity, e.g., each receptor neuron connects only to one projection neuron, i.e., glomerulus.

In more detail, the learning rule for the weights w_{ij} between the projection neuron is specified as

$$\Delta w_{ij} = \gamma p_i p_j - \delta(\sum_{ij} w_{ij} - d) \, . \qquad (10)$$

This is basically a Hebbian learning rule with a penalty term for restricting the weights and two positive and constant learning rates γ and δ. The Hebbian term $\gamma p_i p_j$ gives the network some Hopfield net characteristics. This leads to an attractor network that stabilizes input patterns and reduces noise. The term $(\sum_{ij} w_{ij} - d)$ minimizes the energy or Lyapunov function $(\sum_{ij} w_{ij} - d)^2$. Thus the sum over all w_{ij} is forced have the value d. With initial symmetrical weights, the weights stay symmetrical during learning, thus convergence can be guaranteed.

The learning rule for the weights v_{ij} from sensory neurons to projection neurons is:

$$\Delta v_{ij} = \beta p_i x_j - \hat{\beta}(\sum_{j} v_{ij} - c) - \overline{\beta} \sum_{k \neq i} v_{kj} \qquad (11)$$

with positive constant learning rates β, $\hat{\beta}$ and $\overline{\beta}$. This learning rule has also a Hebbian term $\beta p_i x_j$ and two penalty functions $\hat{\beta}(\sum_j v_{ij} - c)$ and $\overline{\beta} \sum_{k \neq i} v_{kj}$. A Lyapunov function for this learning rule is the following

$$\begin{aligned} E = & - \beta \sum_{ij} v_{ij} p_i x_j \\ & + \frac{\hat{\beta}}{2} \sum_i (\sum_j v_{ij} - c)^2 \\ & + \frac{\overline{\beta}}{2} \sum_j ((\sum_i v_{ij})^2 - \sum_i v_{ij}^2) \, . \end{aligned} \qquad (12)$$

The first term on the right side of Eq.(12) maximizes the network output $(\sum_{ij} v_{ij} p_i x_j)$. The second term $(\sum_i (\sum_j v_{ij} - c)^2)$ restricts the input weights of each projection neuron to a constant parameter c, and the last term enforces each receptor neuron to only make one

synapse, i.e. one weight unequal to zero ($(\sum_i v_{ij})^2 - \sum_i v_{ij}^2$). Note that the weights v_{ij} are restricted to the positive range of $\mathbb{R}$.

The weight updates are done every time step according to the rules:

$$w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij}(t) \quad (13)$$

$$v_{ij}(t+1) = v_{ij}(t) + \Delta v_{ij}(t). \quad (14)$$

Weight initialization at the beginning of the learning procedure can be done so that the restrictions dealing with the sums of weights are fulfilled. For example select all weights w_{ij} and v_{ij} small and randomly so that

$$\sum_{ij} w_{ij} = d \quad (15)$$

$$\sum_{j} v_{ij} = c \quad \text{for all } j \in \{1, \ldots, k\} \quad (16)$$

It should be noted that this learning algorithm is very straight-forward and does not focus very much on the actual implementation in the antennal lobe. In particular, the penalty functions use non-local information for weight adaptation. Several mechanisms are possible for such weight restrictions and normalizations to be implemented more realistically in a local learning rule incorporating local interneurons. Here, however, we explore the behavior of the system under the simple learning rules stated above.

The learning rule for the input weights is closely related to those of the dynamic link architecture proposed by von der Malsburg (Malsburg, 1973, Willchaw & Malsburg, 1976, Konen & Malsburg, 1992), whereas the renormalization differs from application to application. Also, in contrast to von der Malsburg's approach, the normalization in our model does not require a separate computational step, but is rather implemented in a soft way using a penalty function in the learning rule. Activity blobs in the input layer are implicitly given by the shape and migration of odor packages on the antenna. The connectivity in the output layer of our model differs from the topographic static weights in the dynamic link architecture, where the weights w_{ij} reflect the neighborhood of output cells. Since we do not want to have neighboring cells activated at the same time, but cells who encode similar odors, our Hebbian learning rule given in Eq.(10) establishes recurrent links between functional similar units rather than links between units that are close together.

3 Simulation Results

We trained the network using 200 antennal neurons and 100 projection neurons. For modeling antennal ORNs, we used three types of model ORNs which simulate honeybee ORNs (Malaka *et al.*, 1995). We selected quite simple types of ORNs which respond most to one odorant and less or inhibitory to others. During training, odors were presented sequentially. Whereas one odor presentation started with a distribution at an initial center and network performed 20 updates for all neurons and all weights. Then the distribution was shifted slightly and another 20 updates were performed. This procedure was repeated five times, i.e., 100 updates in total were performed during one odor presentation. One learning cycle consists of odor presentations for all odors within the training set.

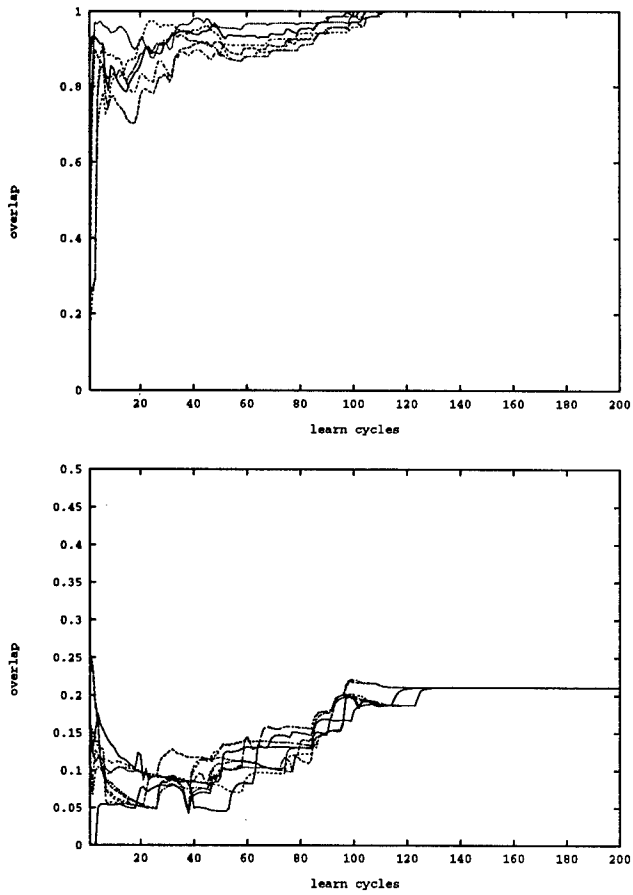


Figure 3: Overlaps ov for odors with different distributions during learning. The upper panel shows the $ov(O_1, O_1, \mu_1, \mu_2)$ for one odor tested with various different distribution centers μ_1, μ_2. The lower panel shows the overlaps for two different odors $ov(O_1, O_2, \mu_1, \mu_2)$ at various distributions.

Figure 3 shows how the network learns to recognize odors independent of their distribution on the antenna. All curves where one odor is tested with two different distributions reach the maximal overlap after 120 learning cycles, i.e., the network recognizes the odor as the same invariant against the particular distribution on the antenna. Whereas the curves for two different odors reach a low value of 0.2, indicating that these two odors are

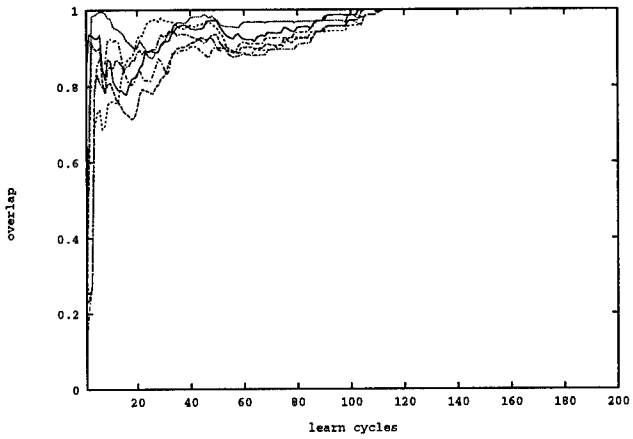


Figure 4: Overlaps ov for one odor at two different concentrations $ov(O_1, 0.5O_1, \mu_1, \mu_2)$ with different distributions during learning.

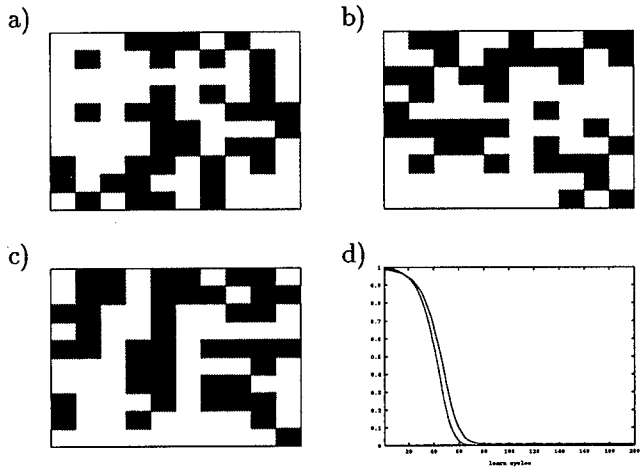


Figure 5: Projection neuron activity maps for (a) an odor O_1, (b) an odor O_2, and (c) their binary mixture O_1+O_2. The grid shows 10×10 projection neuron activities, whereas highly active activities are labeled in black, and low active neurons in white. Plot (d) displays the two terms $\sum_i(\sum_j v_{ij} - c)^2$ and $\sum_j((\sum_i v_{ij})^2 - \sum_i v_{ij}^2)$ from the Lyapunov function for the learning rule (11). Both terms are normalized by their initial value and both converge to 0 during learning.

well distinguishable at any antennal distribution. The network also generalizes and learns to encode odors in a concentration-independent way. As shown in Fig.4, the coding for an odor at two different concentrations yield an overlap of 1 after training.

Thus after learning, the network establishes stable activity maps for each odor, whereas one odor is encoded by the same map at any concentration and distribution. Figure 5a-c shows such emerging activity maps for two odors O_1 and O_2 and their binary mixture $O_1 + O_2$. Moreover, the emerging connectivity of the network fits to the antennal lobe morphology. Due to the learning rule (11), each antennal lobe neuron has only one connection to one projection neuron (alias glomerulus) after learning (Fig.5d).

4 Conclusions

As in many primary sensory systems, the antennal lobes of the honeybee are involved in associative learning (Hammer, 1995, Hammer & Menzel, 1995). VUMmx1, an octopaminergic neuron that represents food reward during associative learning, innervates presumably all AL glomeruli (Hammer, 1993). Serotonergic neurons also terminate within the AL (Rehder *et al.*, 1987). Both serotonin and octopamine can be thought as labeling signals that enable synaptic changes, but it is beyond state of the art to actually measure what kind of synaptic change, i.e., what learning rule, is performed on the antennal lobe.

The learning rule proposed here allows the system to self-organize. It could be shown that this learning paradigm can be extended to the non-topographic mapping of odors. Moreover the self-organizing model establishes an antennal lobe-like network that generates a stable coding of odors irrespective of their spatial concentration distribution on the antenna. It also provides a spatial integrated coding that is independent of the subset of antennal neurons that actually carry the olfactory signal. It has to be noted, however, that the learning algorithm works best with *nice* ORN profiles. Due to the simple Hebbian learning rule, highly non-linear ORN responses cannot easily be processed. If these signals are to be integrated, the learning rule has to be extended and the simple attractor network in the projection neuron layer has to incorporate an advanced learning algorithm that does not crucially depend on orthogonal patterns. Anyway, if the training patters are not generated by actual odor responses, but by some genetic process in the pupal development, it might be the case that strong nonlinearities do not play an important role. In this context, the proposed self-organizing model solves the soft-wiring task of clustering the axons in the antennal nerves and distributing them to the glomeruli in such a way that similar ORNs are collected in one glomerulus.

On the basis of experimental data which is available, however, it is not yet possible to actually decide which learning mechanism is implemented in the antennal lobes. Recent experiments with optical recordings and intracellular recordings during associative learning seem to be a good basis for future investigations, as well as for testing which of these techniques are relevant during learning in the antennal lobe.

On the other hand, the fixed-weight modeling results and the learning rules have to be combined. The study

with fixed-weight networks gives a framework for the type of neuronal elements and the level of abstraction necessary for a realistic modeling of AL properties. In the next steps both approaches have to be integrated.

Acknowledgement

This work was partially supportet by the German Research Foundation (DFG), grant Me672/5-3 (SPP Theorie und Physiologie neuronaler Netze) and the DAAD, grant grant 315/3.

References

Akers, R. P. and W. M. Getz (1992) A test of identified response classes among olfactory receptor neurons in the honeybee worker. *Chemical Senses*, 17(2):191–209.

Akers, R.P. and W.M. Getz (1993) Response of olfactory receptor neurons in honey bees to odorants and their binary mixtures. *J. Comp. Physiol. A*, 173:169–185.

Boeckh, J., P. Distler, K. D. Ernst, M. Hösel, and D. Malun (1990) Olfactory bulb and antennal lobe. In Schild, D., editor, *Chemosensory Information Processing*, pages 201–227. Springer.

Eisenberg, J., W.J. Freeman, and B. Burke (1989) Hardware architecture of a neural network model simulating pattern recognition by the olfactory bulb. *Neural Networks*, 2:315–325.

Földiák, P. (1989) Adaptive network for optimal linear feature extraction. In *Proc. IEEE/INNS Int. Joint Conference on Neural Networks*, volume 1, pages 401–405.

Freeman, W.J. (1987) Simulation of chaotic EEG patterns with a dynamic model of the olfactory system. *Biol. Cybern.*, 56:139–150.

Gelperin, A. and J.F. Hopfield (1988) Differential conditioning to a compound stimulus and its components in the terrestrial mollusc *limax maximus*. *Behavioral Neurosci.*, in press.

Gelperin, A., J.J. Hopfield, and D.W. Tank (1986) The logic of *limax* learning. In Selverston, A.I., editor, *Model Neural Networks and Behavior*, chapter 13, pages 237–261. Plenum Press, New York.

Gelperin, A., D.W. Tank, and G. Tesauro (1989) Olfactory processing and associative memory: Cellular and modeling studies. In Byrne, J.H. and W.O. Berry, editors, *Neural Models of Placticity: Experimental and Theoretical Approaches*, chapter 8, pages 133–159. Academic Press.

Getz, W.M. (1991) A neural network for processing olfactory-like stimuli. *Bulletin of Mathematical Biology*, 53(6):805–823.

Halasz, N. (1989) Morphological basis of information processing in the olfactory bulb. In Schild, D., editor, *Chemosensory Information Processing*, pages 176–190. Springer.

Hammer, M. and R. Menzel (1995) Learning and memory in the honeybee. *J.Neurosci.*, 15(3):1617–1630.

Hammer, M. (1993) An identified neuron mediates the unconditioned stimulus in associative olfactory learning in honeybees. *Nature*, 366:59–63.

Hammer, M. (1995) Local brain injections of octopamine substitute for the us in olfactory associative learning and induce the formation of a lasting memory in the honeybee dependent on injection site. in preparation.

Hendin, O., D. Horn, and J.J. Hopfield (1994) Decomposition of a mixture of signals in a model of the olfactory bulb. *Proc.Natl.Acad.Sci. USA*, 91(13):5942.

Homberg, U., T.A. Christensen, and J.G. Hildebrand (1989) Structure and function of the deutocerebrum in insects. *Ann.Rev.Entomol.*, 34:477–501.

Hopfield, J.J. (1988) Neural computations and neural systems. In Cotteril, R.M.J., editor, *Computer Simulation in Brain Science*, chapter 26, pages 405–415. Cambridge University Press, Cambridge, UK.

Hopfield, J.J. (1991) Olfactory computation and object perception. *Proc. Natl. Acad. Sci. USA*, 88:6462–6466.

Kohonen, T. and E. Oja (1976) Fast adaptive formation of orthogonalizing filters an associative memory in recurrent networks of neuron-like elements. *Biological Cybernetics*, 21:85–95.

Kohonen, T. (1982) Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43:59–60.

Kohonen, T. (1988) *Self-Organization and Associative Memory*. Springer-Verlag, Berlin.

Konen, W. and C. von der Malsburg (1992) Unsupervised symmetry detection: A network which learns from single examples. *Artificial Neural Networks*, 2:121–125.

Li, Z. and J.J. Hopfield (1989) Modeling the olfactory bulb and its neural oscillatory processings. *Biol.Cybern.*, 61:379–392.

Linster, C. and C. Masson (1996) A neural model of olfactory sensory memory in the honeybee's antennal lobe. *Neural Computation*, 8(1):94–114.

Linster, C., C. Masson, M. Kerszberg, L. Personnaz, and G. Dreyfus (1993) Computational diversity in a formal model of the insect olfactory macroglomerulus. *Neural Computation*, 5(2):228–241.

Linster, C., D. Marsan, C. Masson, and M. Kerszberg (1994) Odor processing in the bee: A preliminary study of the role of central input to the antennal lobe.

In Cowan, J.D., G. Tesauro, and J. Alspector, editors, *Advances in Neural Information Processing Systems*, volume 6. Morgan Kaufmann Publishers, San Mateo CA.

Malaka, R. and M. Hammer (1996) Coding capability of an antennal lobe model — the role of input preprocessing and structured interneuron. In Elser, N. and H.-U. Schnitzler, editors, *Proceedings of the 24th Göttingen Neurobiology Conference*, volume 2. Thieme-Verlag, Stuttgart, New York.

Malaka, R., T. Ragg, and M. Hammer (1995) Kinetic models of odor transduction implemented as artificial neural networks – simulations of complex response properties of honeybee olfactory neurons. *Biol. Cybern.*, 73:195–207.

Malaka, R. (1995) Dynamical odor coding in a model of the antennal lobe. In *Proceedings of the International Conference on Artificial Neural Networks (ICANN'95), Paris*, volume 2, pages 503–508. EC&Cie, Paris.

Malsburg, C. von der (1973) Self-organization of orientation sensitive cells in the striata cortex. *Kybernetik*, 14:85–100.

Malun, D. (1991) Inventory and distribution of synapses of identified uniglomerular projection neurons in the antennal lobe of *periplaneta americana*. *The Journal of Comparative Neurology*, 305:348–360.

Martin, J.H. (1991) Coding and processing of sensory information. In Kandel, E.R., J.H. Schwartz, and T.M. Jessel, editors, *Principles of Neural Science*, pages 329–340. Elsevier, New York, Amsterdam, London, Tokyo, 3 edition.

Martinetz, T., H. Ritter, and K. Schulten (1988) Kohonen's self-ornaizing map for modeling the formation of the auditory cortex of a bat. In *Connectionism in Perspective, SGAICO-Proceedings*, pages 403–412, Zürich.

Masson, C. and H. Mustaparta (1990) Chemical information processing in the olfactory system of insects. *Physiol.Rev.*, 70:199–245.

Murlis, J., J.S. Elkinton, and R.T. Cardé (1992) Odor plumes and how insects use them. *Annu. Rev. Entomol.*, 37:505–532.

Murlis, J. (1986) The structure of odor plumes. In Payne, T.L., M.C. Birch, and C.E.J. Kennedy, editors, *Mechanisms in Insect Olfaction*, pages 27–38. Clarendon, Oxford.

Ragg, T., R. Malaka, and M. Hammer (1995) Modeling computational properties of the antennal lobes with artificial neural networks. In Elser, N. and R. Menzel, editors, *Learning and Memory: Proceedings of the 23rd Göttingen Neurobiology Conference*, volume 1, page 24. Thieme-Verlag, Stuttgart, New York.

Rehder, V., G. Bicker, and M. Hammer (1987) Serotonin-immunoreactive neurons in the antennal lobes and suboesophageal ganglion of the honeybee. *Cell Tissue Res.*, 247:59–66.

Ritter, H. and K. Schulten (1986) Topology conserving mappings for learning motor tasks. In Denker, J.S., editor, *Neural Networks for Computing, AIP Conf. Proceedings 151*, pages 376–380, Snowbird, Utha.

Ritter, H., T. Martinetz, and K. Schulten (1991) *Neuronale Netze – Eine Einführund in die Neuroinformatik selbstorganisierender Netzwerke.* Addison-Wesley, Bonn.

Rospars, J.P. (1988) Structure and development of the insect antennodeutocerebral system. *Int.J.Insect Morphol. & Embryol.*, 17:243–294.

Rybak, J. and S. Eichmüller (1993) Structural plasticity of an immunochemically identified set of honeybee olfactory interneurons. *Acta Biologica Hungaria*, 44(1):61–65.

Schild, D. and H. Riedel (1989) Monte Carlo generation of chemosensory maps in the olfactory bulb: Glomerular activity patterns. In Schild, D., editor, *Chemosensory Information Processing*, pages 359–374. Springer.

Schild, D. (1988) Principles of odor coding and a neural network for odor discrimination. *Biophys. J.*, 54:1001–1011.

Schneidermann, A.M., S.G. Matsumoto, and J.G. Hildebrand (1982) Transsexually dimorphism neurones in moth brain. *Nature Lond.*, 298:844–846.

Shepherd, G.M. (1992) Modules for molecules. *Nature*, 358:457–458.

Tolbert, L.P. and J.G. Hildebrand (1981) Organization and synaptic ultrastructure of glomeruli in the antennal lobes of the moth *manduca sexta*: A study using thin sections and freeze-fracture. *Proc. R. Soc. London Ser. B*, 213:279–301.

Vareschi, E. (1971) Duftunterscheidung bei der Honigbiene — Einzelzell-Ableitungen und Verhaltensreaktionen. *Z.vergl.Physiologie*, 75:143–173.

Willchaw, D.J. and C. von der Malsburg (1976) How patterned neural connections can be set up by self-organization. *Proceedings of the Royal Society of London, B*, 194:431–445.

Wilson, D. A. and R. M. Sullivan (1994) Neurobiology of associative learning in the neonate: Early olfactory learning. *Behavioral and Neural Biology*, 61:1–18.

Yao, Y. and W.J. Freeman (1990) Model of biological pattern recognition with spatially chaotic dynamics. *Neural Networks*, 3:153–170.

Some adaptive movements of animats with single symmetrical sensors

Owen Holland and Chris Melhuish
(o-hollan@uwe.ac.uk, cr-melhu@uwe.ac.uk)
Intelligent Autonomous Systems Laboratory, Faculty of Engineering
University of the West of England, Bristol BS16 1QY

Abstract

This paper shows that animats with a single symmetrical sensor, and using only simple two-instant locomotion mechanisms, can move reliably and efficiently towards a point source of stimulation, even in the presence of considerable noise. Some directional sensitivity in the sensor improves performance. Introducing some asymmetry into the algorithms can increase the reliability of finding the source. For environments differing in noise levels, competitive pressures, and availability of resources, different minimal algorithms are required to achieve the best adaptation to each environment.

1 Introduction

Motile organisms adapt to their environment primarily through the appropriateness of their movements. For simple creatures, there are relatively few different situations of interest, and the movements required for good adaptation are easily specified: for example, some local sources of sensory stimulation should be approached, some should be avoided, and some should be kept at a certain optimal distance. Although one might expect that the minimum requirements for the sensory, computational, and motor apparatus necessary for producing movements of these types would be known, this is not the case. Individual species of bacteria, protozoa, and other single celled organisms have been studied, and the functional and physiological basis of the control of movement is in some cases fairly well understood. Some multicellular organisms (e.g. maggots, flatworms, shrimps) sometimes appear to be operating at a very simple level similar to that of the unicellular organisms, and again there is some understanding of the underlying processes. There is a substantial body of classical work devoted to the classification of different types of movement in relation to various types of stimuli; the key original sources are Loeb (1913), Kühn (1919), and Fraenkel and Gunn (1940, 1961), and an excellent modern summary accessible to non-biologists like ourselves is given by Schöne (1984). However, there appears to be no systematic comparison of the abilities, efficiency, and effectiveness of various general mechanisms under different conditions of environmental stimulation and noise, and under various constraints on the available sensory, computational, and motor resources. This paper is an attempt to make such a comparison for a limited subset of agents, in order to enable the theoretical basis of adaptation through movement to be better understood.

The animats examined will operate in a planar environment containing a single fixed source of stimulation. They will have a longitudinal axis, and will be capable of movement forwards in the direction of this axis, and also of rotating the axis in either direction. For simplicity, a given segment of movement will consist of a turn followed by a forward movement; all trajectories will therefore consist of concatenated linear segments. Most importantly, each animat will be limited to a single sensor element, with a single field of view which is continuous and symmetrical about the longitudinal axis. The instantaneous output of the sensor will be equal to the instantaneous strength of the stimulation due to the source, subject to some allowance for noise. This model is easy to simulate, and will also be relatively easy to implement on a wheeled robot, which is one planned outcome from the study.

It might be objected that such a model is too simple to be really interesting. After all, Braitenberg (1984) disposed of the single sensor Vehicle 1 in less than three pages, moving on to the two sensor vehicles which seem to have been adopted by the SAB community as the minimal interesting robot designs. Vehicle 1 is only capable of active movement forwards; any lateral variation comes from physical interactions with the environment such as friction (and presumably collision). In biological terms, Vehicle 1 exhibits direct orthokinesis (Schöne, 1984), the increase of linear speed with increasing intensity of stimulation, a phenomenon observed in creatures such as fruit flies, mosquitoes, and flatworms. Although there are wheel arrangements where the kinematics can produce different trajectories from different values of a single signal (for example, separate steering and driving motors) these vehicles are only marginally more interesting than the basic Vehicle 1. A simple enhancement of Vehicle 1 is the provision of separate turn and move outputs, one being controlled by the sensor, and the other being controlled by some fixed or random output. This allows the vehicle to cover the plane in a way which does not depend on friction

or collision, but which enables the vehicle to spend more time in regions of low stimulation (by slowing down, or by turning more frequently), and to move rapidly through regions of high stimulation using the opposite reactions. (In his "Biological Notes on the Vehicles" Braitenberg points out the connections between Vehicle 1 and classical and recent work (Koshland, 1980) on simple movement control). However, a much richer set of behaviours can be produced if the output of the single sensor can be processed to yield separate outputs for turning and linear progress. In Braitenberg's terms, this would constitute a hybrid vehicle with the single sensor of Vehicle 1 and the two motors of subsequent vehicles. It would certainly be interesting to examine the abilities of such agents when the relationships between sensory input and each motor output were varied within the categories identified by Braitenberg (positively monotonic, negatively monotonic, continuous, and discontinuous) but the focus here will be on a different and arguably simpler class of mechanisms.

Braitenberg's simple two-motor vehicles have two symmetrically arranged sensors. (The sensors as originally described are not directionally sensitive, although implementations using light as the stimulus typically use directional sensors). The vehicles achieve their functionality by sensing the stimulus field strength at these two separated points; the behavioural outcomes are a function of the relative and absolute strengths of the instantaneous inputs to the two sensors. However, this is not a strategy which can be used when the spatial separation of the sensors is insufficient to give different readings. Very small organisms such as bacteria cannot achieve any useful separation of sensors; even larger organisms such as insects with sensors mounted on the tips of long antennae to maximise their separation are unable to resolve differences in very weak chemical fields. The most common solution to this problem is the acquisition of spatially separated readings by moving the body of the organism between the sites of the readings. This is a process which clearly involves the temporal integration of successive sensor readings. The readiness with which such solutions are used in nature can be seen in the bee (Martin 1964). In a strong chemical field, a bee undertakes gradient ascent by using conventional osmotropotaxis, turning towards the antenna which is most strongly stimulated. If it is deprived of one antenna, it will move the other in a zig-zag path, by moving the antenna and also its whole body; the trend of its body movements will be to turn towards the side of the zig-zag on which the antenna receives the higher stimulation. An intact bee in a very weak field will move its whole body on a wide zig-zag path, apparently following a similar rule.

The simplest temporal integration arrangement is known as a two-instant mechanism (Feinleib, 1980), in which the motor output is derived from the current sensor input and the immediately previous sensor input. Only two-instant mechanisms will be considered in this paper, and they will be of the simplest possible kind: if the sensor input has increased, a particular combination of turn and move will be carried out; if the input has decreased, a different combination will be used..

The paper is organised as follows. First, the agent is defined and a general form of two-instant mechanism is proposed, the relevant aspects of the environment are identified and formalised, and performance measures are defined. Second, details of the simulation are given. Third, the performance of a number of mechanisms is described and analysed. Finally, the significance of the results for our understanding of adaptive movement, both natural and artificial, is discussed.

2 The agent and the environment

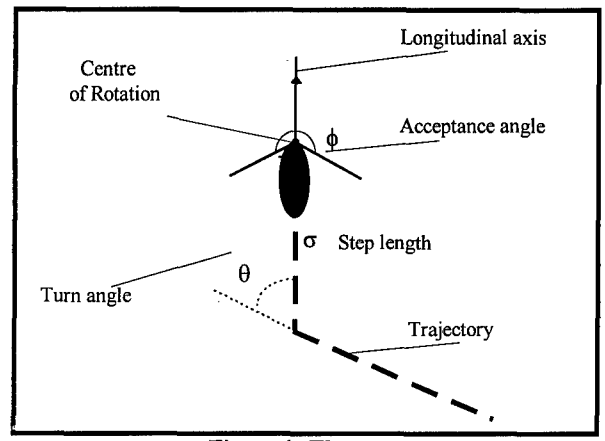


Figure 1: The agent

The agent is modelled as in Figure 1. It has a centre about which it can turn at every time step; θ, the angle of turn, is expressed in relation to the previous direction of the axis. It can move forwards in the direction of the axis after each turn; the distance moved is the step length σ. Rotation and translation are complete by the next time step, and the agent is at rest before the next rotation and translation - it has no linear or angular momentum. The closest biological analogue would be a creature with a saltatory mode of movement, such as a flea or shrimp. At each time step, a new turn angle and step length are generated from the current and previous sensor readings.

The sensor is radially sensitive to the line-of-sight stimulation produced by the source. It has equal sensitivity over some continuous acceptance angle ϕ which is symmetrically arranged across the midline, and which includes the forwards direction. This arrangement was selected because it gives minimal information about the lateral orientation of the source with respect to the agent's axis. It is worth noting that simple creatures such as protozoa which possess some distal sensing ability often have a single asymmetrical sensor. For example, *Euglena gracilis* has an asymmetrically positioned light-sensitive pit (the stigma) backed by a layer of opaque pigment (the

screening organelle). It can therefore only detect light on one side. Although *Euglena*'s motion (in three dimensions - it follows a helical course) is quite well understood, it is not clear whether there are any specific benefits obtainable from asymmetrical sensors in general. One aim of the work described here is to provide a baseline against which the contribution of asymmetry can be evaluated.

An acceptance angle of 360° means that the sensor is equally sensitive in all directions and therefore can supply no instantaneous information about the orientation of the agent in relation to the source. This corresponds to sensing the local concentration of a chemical, or local temperature, or any other non-directional information. If such local variables are distributed in a way which depends only on their distance from some point, their sensing may be modelled in terms of a radially sensitive 360° sensor using line of sight stimulation from a source at that point. This allows us to model both directional and non-directional sensing with essentially the same sensor model.

The basic two-instant rule can be expressed in a simple canonical form. Let the current sensor reading be Sc, and the previous one be Sp. Then it is:

if (Sc>Sp) **then** turn θa and move σb

else turn θc and move σd

where θa, σb, θc, and σd are randomly and independently selected from some distributions A, B, C, and D. The space of possible rules of this type is therefore the space defined by the various combinations of distributions possible. It is of course huge; however, this study will use only uniform distributions as convenient exemplars of unimodal distributions. The only relevant agent parameter is the sensor acceptance angle.

The canonical form can be adapted to give a movement pattern which seems very common in bilaterally symmetrical creatures: turning in alternating directions. Such motion often results from non-computational factors. For example, a creature which has just moved by contracting the muscles on one side of its body will usually produce the next movement by contracting the muscles on the other side of its body. To the extent that each movement has a turning component, the turns will naturally alternate in direction. However, it is convenient to express this capacity for alternation in computational terms. Schöne notes: "The regular side-to-side alternation is usually based on an idiothetic program that instructs the animal to turn left following a right turn and vice versa. The stimulus directly influences only the angle of the turn or the distance between turns, and not the side towards which the animal turns." (Schöne, 1984) In our more general approach, it is certainly possible for the direction of turn to be specified by the movement rules, and so it may be that one half of the rule may specify alternation, and the other a fixed or randomly chosen direction.

The environment will be limited to a bounded arena with a source of stimulation placed at the centre. The intensity of stimulation will vary with the radial distance from the source. However, in the real world, noise must obviously be taken into account. Noise can arise in a number of ways: from variation within the source; from the characteristics of the transmission medium; from background levels of stimulation; and from the sensory process itself. These factors can combine to yield an effective signal to noise ratio, where the signal is considered to be an idealised signal from a hypothetical steady source. This ratio will typically decrease as the distance from the source increases (unless of course the only noise comes from the source itself, which is unlikely). A convenient way of modelling this qualitatively is to consolidate all variation into a Gaussian background noise function which is uniform over the arena; the sensed signal strength at a location is obtained by summing the background noise with the idealised signal.strength. A suitable choice of mean background noise, arena size, and stimulus decay function can yield a situation in which there is almost perfect information close to the source, and much less useful information at the periphery of the arena. This means that it should be possible for even good algorithms to fail to reach the source within a certain time if released in the badlands at the periphery.

3 Performance measures

With a static source, there are a number of performance measures available. Agents released a certain distance from the source, and in an appropriate distribution of initial orientations, can conveniently be assessed in terms of the probability of reaching the source, the distance travelled (normalised by expressing it as a multiple of the Euclidean distance to the source), the energy used, or the number of time steps taken to reach the source. Although with many algorithms the last three metrics will be strongly correlated with one another, the differences between them may be significant when comparing algorithms. For instance, in a competitive and crowded environment where food items appear suddenly, it may be the time taken to reach an item which determines whether a given item is consumed by one agent or another; a slow and steady algorithm will then produce less food than a much faster but less reliable type. In an environment where resources are scarce and there is no competition, reliability and energy efficiency will be most important.

In order to assess the probability of reaching the source, agents should be released in a position from which it is possible to succeed through the merits of the algorithm - that is, a position in which at least some useful information from the source is present - but from which failure is also possible. This is achieved by releasing the agents a short distance inside the periphery. If they cannot use the information to move in the direction of the source, there is a good chance that they will migrate to the periphery; hitting

the periphery is counted as a failure. The other performance measures are all straightforward.

One problem is that many algorithms using restricted ranges for the turn angle can approach a point source quite well but circle round it at some fixed radius, rather than approaching progressively closer. This does not mean that such algorithms are useless. Many species use different mechanisms depending on the strength of the sensory stimulation or distance from the source, and a mechanism's usefulness may consist in the fact that it moves the organism close enough to the source to allow some other mechanism to come into operation. In the case of the flatworm *Planaria*, Schöne notes: "At the margin of the (food) gradient, flatworms move in wide loops....The closer the planarians get to the source, the higher the animals' turning rate. Their speed of motion also increases (=orthokinesis). Usually direct orientation processes (=taxis) take over close to the source" (Schöne, 1984). Now the source must clearly have some dimension, even if the agent does not. In practice, with the parameter settings explored in this paper, the diameter of any terminal circular movements is always less than 10% of the arena diameter. The source was therefore given a diameter of 10% of the arena diameter; this enabled all algorithms to be compared on a similar basis. However, it is of course worth noting which algorithms can reach much smaller targets; these would not require any secondary mechanism for the final capture of the source object.

4 Simulation details

The arena was modelled as a circular area of 1200 units diameter, with the source at the centre. Agent movements and turns were both represented and calculated as double precision floating point quantities. On each trial, the agent was started 75 units from the periphery in a random orientation with respect to the source. Each algorithm was tested under three levels of background noise (zero, 5%, and 10% of the maximum signal strength) and for four acceptance angles (90°, 180°, 270°, and 360°). Where algorithms were parametrised, a number of parameter settings were tried. For each condition, 25 trials were run; the proportion of successful trials was recorded, and performance indices were calculated from the successful trials.

5 Results

4.1 Random turn algorithms (*E.coli* type)

In the model formulated by Koshland (1980) and tested in simulation by Marken and Powers (1989), *E. coli* moves in a series of straight line segments separated by random tumbles in which the direction after tumbling is uncorrelated with the direction before tumbling. Tumbling occurs when the leakily integrated sum of the increases and decreases in stimulation since the last tumble exceeds some threshold value. We investigated a simplified two-instant version of this type of algorithm, **Rule 16**:

if (Sc>Sp) **then** Rotate through +/-random(5°)
Move forward m units +/-random(5%m)
else Rotate through +/-random(180°)
Move forward random(5%m)

The step length parameter m was set to 6.

The results are shown in Table 1, and are surprisingly rich. The performance metrics used were probability of success, and distance traversed (expressed as a multiple of the minimum distance). With the 360° acceptance angle, corresponding to sensing a chemical concentration as does *E. coli*, the algorithm traverses only 1.66 times the minimum distance in the noise-free condition, and is always successful. While not as efficient as Marken and Powers' more complex algorithm, this is surprisingly good. With 5% noise, the proportion of successful trials is only 36%, and the distance travelled is up to 18.21; at 10%, only 20% of trials succeed, and they involve 16.03 distance units. Marken and Powers did not explore the effects of noise on their model; from our data, we have to conclude that significant noise levels are capable of significantly degrading the performance of our model.

One possible weakness of Rule 16 is that the step length for the (else) condition is very short, and it is possible that in a noisy environment this will yield a less reliable sample point than would a larger step. We investigated the performance of a variant (Rule 0) which makes the step length in the (else) condition equal to that in the (if) condition. The results are shown in Table 2. They are slightly worse in the noise-free condition, which might be expected since directions are not 'checked' for goodness before executing the full step length, but they are still 100% successful. With 5% noise, the success rate is lower than with Rule 16, but the distance travelled is also lower. However, at 10% noise, the success rates are equal, and Rule 0 again travels lower distances. This indicates that for high noise values, Rule 0 might be preferable to Rule 16.

Turning to the Rule 16 results for sensors with acceptance angles less than 360°, we first observe that all are considerably more reliable than 360° under all noise conditions. This is a blanket endorsement of the advantages of some directional sensitivity. In the noise-free condition, the distance travelled for the 90° acceptance angle is only 11% greater than the minimum possible. The animat achieves this efficiency because it only moves a large step if it is pointing within 45° of the source, and, by simple geometry, every such movement is guaranteed to bring it between m and $m/\sqrt{2}$ closer to the source. The 180° and 270° conditions are both still reasonably good, but are worse than the 360° condition. In the 180° condition, long steps will always be towards or at worst tangential to the source. However, at 270° it is possible for large steps to be

acceptance angle	no noise % success	no noise total distance	5% noise % success	5% noise total distance	10% noise % success	10% noise total distance
90°	100	1.11	100	3.68	76	5.54
180°	100	1.84	100	2.90	96	3.80
270°	100	1.93	100	3.34	92	4.91
360°	100	1.66	36	18.21	20	16.03

Table 1: Rule 16 performance

acceptance angle	no noise % success	no noise total distance	5% noise % success	5% noise total distance	10% noise % success	10% noise total distance
90°	100	1.36	84	7.42	52	8.89
180°	100	1.92	92	5.42	64	6.21
270°	100	1.91	84	6.41	72	7.65
360°	100	1.91	20	11.49	20	12.35

Table 2: Rule 0 performance

taken away from the source. This is not possible for the 360° condition, because any step away from the source will result in a decrease in the signal strength, and will therefore produce a small step. This probably explains why 270° is the worst performer under conditions of no noise.

The addition of noise changes the picture. It severely degrades information about source location obtained by small steps, but information obtained from the agent's orientation is more robust. For both noise levels, the 180° condition scores best, perhaps because it gains useful information from orientation for about half the time, whereas the 90° condition does so only for a quarter of the time, other things being equal. The 270° condition scores better than the 90°, presumably showing that the benefits of capturing more information under noisy conditions outweigh the drawback of making occasional large steps away from the source.

Rule 0 scores worse than Rule 16 under almost all restricted acceptance angle conditions, and achieves lower success rates. Interestingly, the ordering of the scores is exactly the same as in Rule 16. Figure 4a shows a typical trajectory for Rule 16.

Alternating turn algorithms (*Planaria* type)

The basic idea behind all alternating turn algorithms is that the agent should turn towards the side that is more strongly stimulated. This can easily be done by increasing the turn angle if the sensor reading has decreased. A rule which expresses this, and incorporates some variability into the movements, is **Rule 28:**

if previous rotation was L **then** Rotn=R **else** Rotn=L
if (Sc>Sp) **then** Rotate through d° +/-random(5°) in direction Rotn
Go forward m units +/-random(5%m)
else Rotate through d°+random(20°) in direction Rotn
Go forward m units +/-random(5%m)

Note that the random(20°) in the last condition is equivalent to 10° +/-random(10°). This extra variability is useful in certain conditions, but is not essential to the algorithm.

The turn angle parameter d is a major influence on performance, and so the range of possibilities is covered in 20° increments (20° through 160°). The step length m was again set to 6 units. The results of trials under three noise conditions are shown in Tables 3a-3f. (A tabular form is used because of space constraints.)

Consider first the 360° acceptance angle case. For no noise, the success rate rises from 48% at 20° turn angle to 100% at 140° and 160°. However, the average distance travelled on a successful trial falls from a figure of 3.8 at 20° to a minimum of 1.7 at 80°, and then rises to a maximum of 6.1 at 160°. The behaviour of this algorithm is clearly more complex than those previously examined.

A necessary concept in understanding alternating turn algorithms is that of the trend line - the line describing the general direction of movement of the agent when the zig-zags are filtered out (see Figure 2). Such a line can be constructed by joining the mid-points of each step. For a given turn angle θ, the shortest path to the source is a zig-zag with the trend line passing through the source; simple trigonometry shows that the length of the trajectory is greater than the length of the trend line (the Euclidean distance) by a factor of $1/\cos(\theta/2)$. The control part of the algorithm can improve the score only up to this level. The best possible score for each condition is shown in Table 4.

Turn angle	40°	20°	60°	80°	100°	120°	140°	160°
Factor	1.06	1.02	1.15	1.31	1.56	2.00	2.92	5.76

Table 4: Minimum path length correction factor

	20	40	60	80	100	120	140	160
90	44	56	48	60	68	68	84	96
180	52	76	68	84	100	100	100	100
270	72	88	92	100	100	92	100	100
360	48	68	68	72	88	88	100	100

Table 3a: Rule 28, no noise, % success

	20	40	60	80	100	120	140	160
90	1.2	1.2	1.3	1.4	1.8	2.5	4.1	12.1
180	3.1	2.3	1.8	1.7	1.9	2.2	3.2	6.2
270	4.5	2.4	1.9	1.9	1.9	2.3	3.2	6.2
360	3.8	2.3	1.8	1.7	1.9	2.3	3.2	6.1

Table 3b: Rule 28, no noise, distance travelled

	20	40	60	80	100	120	140	160
90	52	52	48	60	80	84	88	92
180	60	68	88	76	88	100	100	100
270	20	52	72	100	100	100	96	100
360	12	16	16	12	24	24	32	36

Table 3c: Rule 28, 5% noise, % success

	20	40	60	80	100	120	140	160
90	1.2	1.2	1.3	1.5	1.8	2.4	3.7	8.7
180	2.4	1.7	1.6	1.7	1.9	2.5	3.6	7.9
270	3.9	2.3	3.1	3.2	2.8	3.0	4.1	8.6
360	2.1	1.3	2.3	1.9	2.3	4.1	5.7	13.5

Table 3d: Rule 28, 5% noise, distance travelled

	20	40	60	80	100	120	140	160
90	44	56	60	48	60	68	76	88
180	56	68	72	60	84	72	100	100
270	16	32	48	80	76	92	96	100
360	16	24	16	12	28	12	28	24

Table 3e: Rule 28, 10% noise, % success

	20	40	60	80	100	120	140	160
90	1.2	1.2	1.4	1.5	1.8	2.5	3.8	9.0
180	2.1	1.9	1.7	1.8	2.0	2.5	3.7	8.0
270	1.4	2.5	3.4	2.8	3.0	3.5	4.3	9.1
360	1.2	1.3	2.3	2.6	2.4	3.1	7.3	11.7

Table 3f: Rule 28, 10% noise, distance travelled

For an agent which has a mode of locomotion which necessarily produces some alternation, these factors should be taken into account. However, if the agent is capable of moving in a straight line, the comparison with the Euclidean distance is still appropriate.

With the previous algorithms, the success rate and the performance metric were usually positively correlated. However, there are some negative relationships in the data from Rule 28. Since the trials use agents starting in various random orientations; some orientations lead straight to the source, and others require the agent to turn round in a region of poor signal to noise ratio before it can move

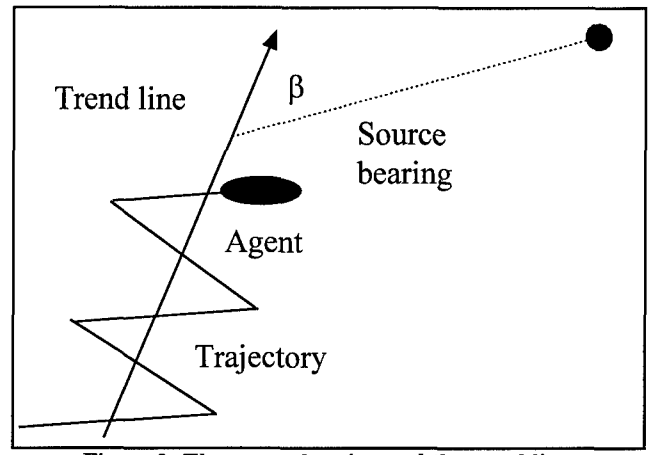


Figure 2: The source bearing and the trend line

towards the source. The trajectories for successful runs from the more difficult orientations tend to be longer, other things being equal, because the minimum possible path is longer, and also because the higher uncertainty in peripheral regions of the arena produces more meandering. If a change in a parameter causes runs from the more difficult orientations to hit the periphery and fail, the mean distance travelled for the successful runs may be decreased, even if the parameter change makes the successful runs travel slightly further. It does not necessarily mean that the algorithm has become faster but less reliable, which is an inference that could be drawn from the results at first sight, for example when comparing the 100° and 80° conditions. This could be resolved by plotting initial orientation as a variable; data are insufficiently detailed at present.

The noisy conditions present a picture which is in parts ambiguous for the same reason. Increasing noise tends to reduce the success rate; only in those cases when the distance travelled goes up as the success rate goes down can one deduce from the figures that the performance of the algorithm is probably becoming degraded in all respects.

For 360° with no noise, Rule 16 is clearly better than Rule 28, showing a higher or equal success rate and shorter total distances. However, with 10% noise, Rule 28 is superior in both success rate and distance travelled for most turn angle values. This is remarkable, given that some of these turn angles inevitably bring with them high minimum distance scores. Again, for other acceptance angles, Rule 16 is always best with no noise, but with noise the best performances of Rule 28 beat Rule 16.

Because of the relatively good performance of Rule 28 under noisy conditions, it will be worth examining its mode of operation in some detail. At any time, the source has a certain bearing angle β in relation to the current trend line. (See Figure 2) If the source is ahead and to the right of the trend line, then taking a step to the right will always produce an increase in S in a noise-free situation. However, taking a step to the left under those conditions will produce

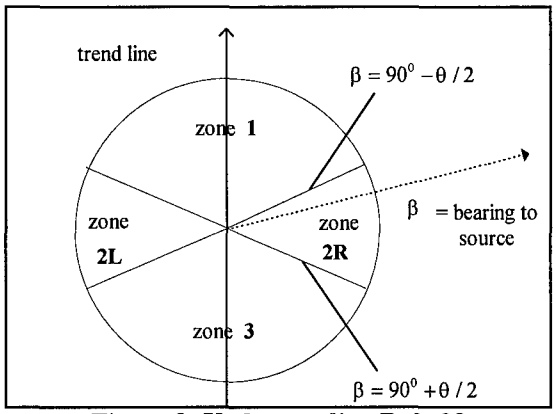


Figure 3: Understanding Rule 28

an increase only when $\beta<(90-\theta/2)$, where θ is the turn angle in the (if) condition. This remains true as β is increased, and the source is behind and to the right, until the value $\beta=(90°+\theta*/2)$ is exceeded, when both left and right steps produce a decrease in S. ($\theta*$ is the turn angle produced by the (else) condition). When the source is on the left, the same changes occur with increasing β. Figure 3 shows these relationships diagrammatically. Now when β is in zone 1, the agent zig-zags forward symmetrically. However, in zone 2L, it turns through $\theta*$ on the left step, and θ on the right step, turning the trend line left towards the source; similarly, in zone 2R, the trend line turns right towards the source. In zone 3, the agent again zig-zags symmetrically forward, moving away from the source, but at a lower rate of progress per step than in zone 1, because the turn angle is set at $\theta*$.

An agent operating in zone 1 will move along its current trend line (with small deviations due to the random elements in the determination of turn angle and step length). This causes β to increase, and eventually the agent will move into zone 2 when $\beta>(90°-\theta/2)$. The trend line will then be rotated by an amount $(\theta*-\theta)$. This will move the agent back into zone 1 operation. However, the subsequent events are determined by the size of $(\theta*-\theta)$ in relation to β. For $(\theta*-\theta)<\beta$, the new trend line will still have the source on the same side as before. The agent will move along the trend line until it again moves into zone 2 operation; again the trend line will be rotated, and again the source will be on the same side. The agent will be moving along a piecewise linear logarithmic spiral, which is of course a well known biological form. As it approaches the source, a point will be reached where the trend line is rotated at the maximum rate, every other step, and the agent will circle the source. If the source was initially on the left, the spiral and the circus movement will be anticlockwise; if on the right, both will be clockwise. Figures 4b and 4c show typical spiral approaches; the circus movement is not seen because of the size of the source.

For $(\theta*-\theta)>(90°-\theta/2)$, the trend line is rotated in such a way that the source is now on the other side of it. The agent will follow the trend line until again β increases to $(90°-\theta/2)$, and will then rotate the trend line in the opposite direction. It will therefore home in on the source by zig-zagging between two converging logarithmic curves; it will not circle the source. Figure 4d shows a typical approach of this type.

When an agent is on a trend line lying well within zone 2, it will initially rotate the trend line every other step, describing a circular course of the same radius as the circus movements. At some stage the trend line will reach zone 1, when one of the behaviours described above will take over.

An agent started in either zone 1 or 2 will therefore turn at the maximum rate towards the source (see Figure 4e) and will then spiral in to a circular path near the source, or zig-zag in to reach the source. This is impressively efficient. The problems with Rule 28 lie in zone 3, because an agent in zone 3 will move away from the source, increasing β closer to 180°, and thus remaining in zone 3 unless the random components of the movement bring it by chance to the boundary with zone 2. If this boundary is reached by chance, the source will be reached by the process described above. The extra noise in the movement in the (else) condition of Rule 28 was introduced in an attempt to ameliorate this problem. Observation of the runs of Rule 28 show that agents in an initial orientation placing them in zone 3 simply continue to the periphery, recording a failure, unless the noise levels by chance rotate them enough to reach zone 2(see Figure 4f).

A little reflection shows that the cause of the problem is the symmetry of the algorithm. The zone 4 problem can be eliminated by making the (else) part of Rule 28 asymmetrical - in other words, always turning in the same direction. This was implemented by forcing the (else) condition in Rule 28 always to turn right in **Rule 29**:

if previous rotation was L **then** Rotn=R
else Rotn=L
if (Sc>Sp) **then** Rotate through d° +/-random(5°) in direction Rotn
Go forward m units +/-random(5%m)
else Rotate R through d° +random(20°)
Go forward m units +/-random(5%m)

In terms of the zones calculated for Rule 28, this change does not affect zones 1 and 2R. However, in zones 2L and 3, the agent will make two or more right turns in succession. For small turn angles, this will produce tight circles, which will bring the agent into zone 2R or zone 1, when it will behave again as in Rule 28. For larger turn angles, an abrupt course change, or series of changes, will result. Again, when the agent is in an orientation corresponding to zones 2R or 1, it will behave as in Rule 28.

The effects of the change in the algorithm can be seen in the results plotted in Tables 5a - 5f, with parameters as for Rule 28. With no noise, the success rate is 100% for all conditions; the average distance moved per successful run is much the same or slightly greater than for for Rule 28, indicating that the benefit lies in salvaging previously

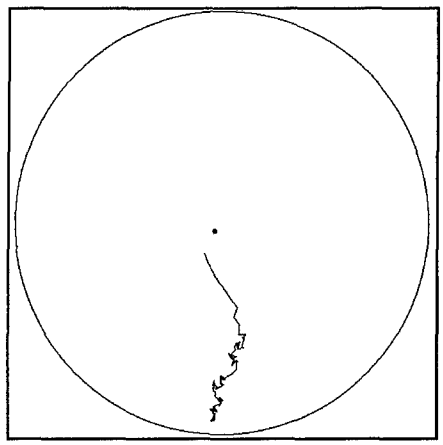


Figure 4a: Typical trajectory from Rule 16

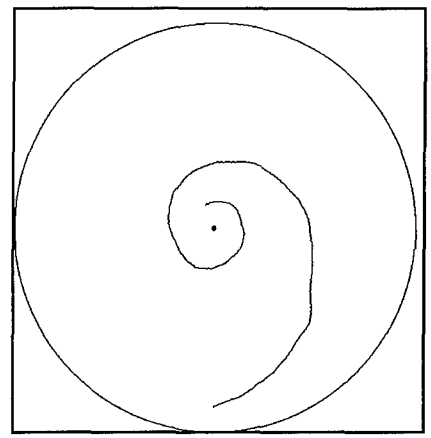


Figure 4b: An anticlockwise logarithmic spiral from Rule 28

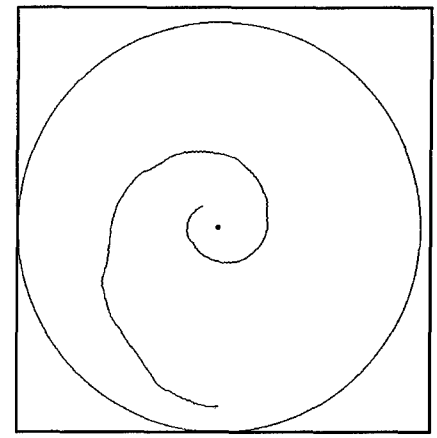


Figure 4c: a clockwise logarithmic spiral from Rule 28

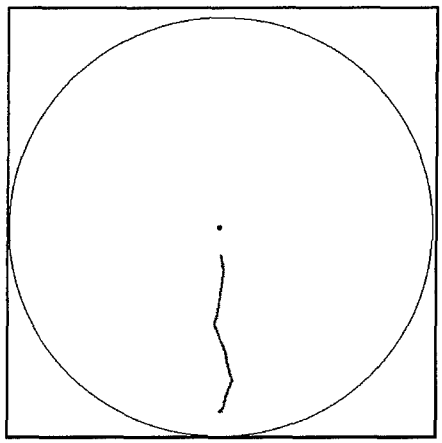


Figure 4d: A zig-zag approach from Rule 28

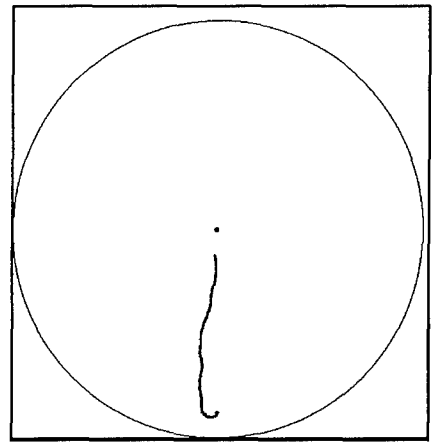


Figure 4e: a turn at the maximum rate from Rule 28

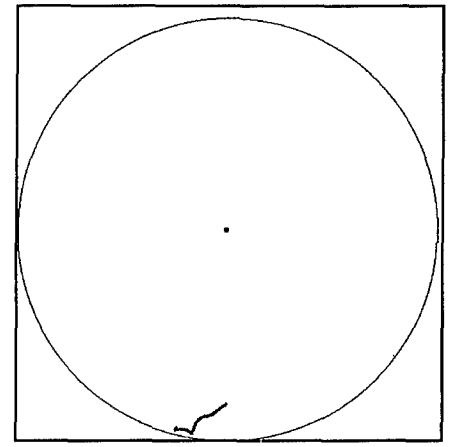


Figure 4f: a failure from Rule 28

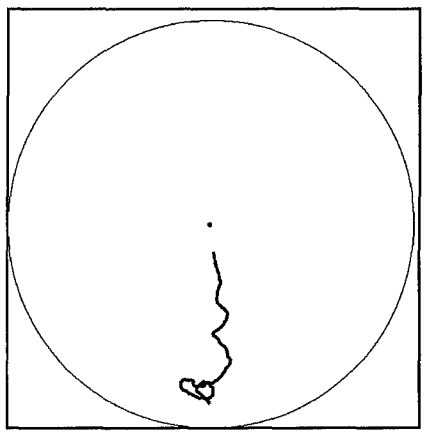


Figure 4g: A natural looking trace from Rule 28 in a noisy environment

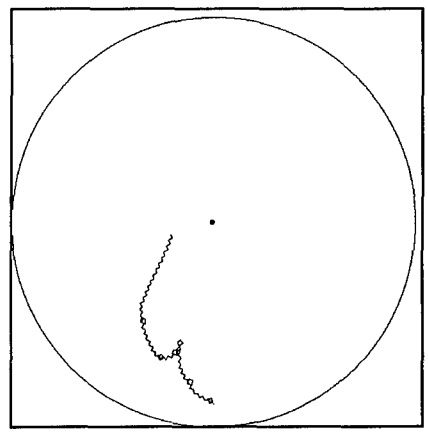


Figure 4h: A typical trajectory from Rule 29

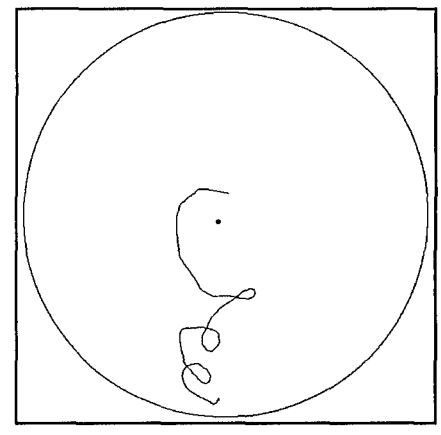


Figure 4i: A trajectory from Rule 31

	20	40	60	80	100	120	140	160
90	100	100	100	100	100	100	100	100
180	100	100	100	100	100	100	100	100
270	100	100	100	100	100	100	100	100
360	100	100	100	100	100	100	100	100

Table 5a: Rule 29, no noise, % success

	20	40	60	80	100	120	140	160
90	1.2	1.2	1.4	6.4	4.3	6.1	7.5	29.2
180	3.0	2.0	1.7	1.7	1.8	2.2	3.1	6.6
270	3.2	2.4	1.9	1.8	1.9	2.2	3.1	6.5
360	3.3	2.2	1.8	1.7	1.9	2.2	3.1	6.5

Table 5b: Rule 29, no noise, distance travelled

	20	40	60	80	100	120	140	160
90	64	88	100	0	100	100	100	100
180	72	84	100	92	100	100	100	100
270	32	0	0	100	96	88	100	100
360	40	44	44	44	48	52	64	48

Table 5c: Rule 29, 5% noise, % success

	20	40	60	80	100	120	140	160
90	13.9	6.0	3.9	N/A	5.6	6.1	6.7	13.0
180	13.1	9.9	5.9	8.7	5.9	4.7	5.0	11.1
270	18.8	N/A	N/A	8.0	17.9	34.2	9.4	12.6
360	15.1	15.6	22.3	21.0	29.6	39.4	49.2	48.6

Table 5d: Rule 29, 5% noise, distance travelled

	20	40	60	80	100	120	140	160
90	64	68	76	28	92	100	100	100
180	60	72	84	76	96	100	100	100
270	16	4	0	96	48	84	100	100
360	36	52	28	32	44	28	36	40

Table 5e: Rule 29, 10% noise, % success

	20	40	60	80	100	120	140	160
90	18.6	8.2	5.7	48.8	8.0	7.9	9.5	13.8
180	18.5	12.3	9.2	12.2	8.5	7.9	7.0	11.7
270	20.3	55.0	N/A	11.4	26.5	40.7	16.3	14.0
360	25.0	19.9	39.7	29.8	36.4	41.6	54.0	54.0

Table 5f: Rule 29, 10% noise, distance travelled

unsuccessful orientations rather than in making the trajectories from good orientations shorter. The best performance for 360° acceptance angle is about the same as that for Rule 16. With 5% and 10% noise, Rule 29 is generally more successful for 360° acceptance angle than Rule 28 and Rule 16, but travels much further on successful runs. However, there are certain combinations of acceptance angle and turn angle (270° acceptance angle and 60° turn angle) which are spectacularly unsuccessful; moreover, these values can be adjacent to some of the best performers (270° acceptance angle and 80° turn angle). The algorithm may therefore not be robust against parametric variation or drift. For acceptance angles of less than 360°, Rule 29 is generally more successful than Rule 28, but often less successful than Rule 16, and tends to travel further than both on successful runs. Of the rules discussed so far, Rule 29 should therefore be preferred for locating sources of chemical stimulation in a noisy non-competitive environment, and would also perform well in a competitive environment with no noise. However, Rule 28 would be preferred for chemical sensing in a noisy competitive environment.

Figure 4g shows a typical and very natural looking trace of a Rule 28 agent in a noisy situation. Figure 4h shows a typical trace of a Rule 29 agent.

Biased non-alternating algorithms (*M. speculatrix* type)

Since it appears that the success rates of alternating algorithms in noisy conditions can be improved by adding asymmetry, it may be useful to examine the effects of adding bias to the non-alternating Rule 16. Rule 16 was therefore modified by constraining the turn in the (else) condition from being completely random to being a small but variable turn to the right, in **Rule 31**:

if (Sc>Sp) **then** Rotate through +/-random(5°)
Go forward m units +/-random(5%m)
else Rotate right through 10°+random(25°)
Go forward random(5%m)

Table 6 shows the results for m=6. With no noise, it is 100% successful, but travels much further than Rule 16 for 360° acceptance angle. With 5% and 10% noise, it is always inferior to Rule 16 for limited acceptance angles. However, for 360° acceptance angle, it is a good overall performer in noisy conditions, achieving higher success rates than all except one parameter combination for Rule 29, and producing trajectory lengths much shorter than Rule 29, but rather longer than Rule 28. Figure 4i shows a typical trajectory.

Rule 31 is of particular interest because it is very similar to the algorithm used on the first behaviour based autonomous mobile robot, the tortoise, designed and built by Grey Walter in 1949 (Walter, 1950, 1953; Holland, 1996). The tortoise was equipped with a single driving wheel which could be rotated clockwise relative to the chassis by the steering motor. A directional photodetector aligned with the steering apparatus controlled the steering and drive motors. Releasing the tortoise (or one of the replicas recently constructed in our laboratory) with the driving wheel aligned with the chassis axis, and with a lamp some distance away on the right of the tortoise would produce a smooth curving trajectory ending with the tortoise next to the lamp. However, if the lamp was on the left, the tortoise would execute repeated small circles to the right, only gradually moving towards the lamp in a sequence of manoeuvres reminiscent of precession. Figure 4i is strikingly similar to some of Grey Walter's original trajectory drawings (Walter, 1950).

acceptance angle	no noise % success	no noise total distance	5% noise % success	5% noise total distance	10% noise % success	10% noise total distance
90°	100	1.15	36	22.17	28	27.46
180°	100	3.80	72	13.54	60	19.51
270°	100	3.95	72	10.66	56	16.67
360°	100	4.06	60	17.37	44	24.35

Table 6: Rule 31 performance

6 Conclusions and future work

The class of minimal two-instant movement control algorithms examined in this paper show surprisingly good performance in finding point sources of stimulation reliably and efficiently, in noisy and quiet conditions. With a single symmetrical sensor, performance is greatly improved by having some directional sensitivity. Introducing some asymmetry into the algorithms can improve the reliability of finding a source, though at the cost of reducing the efficiency in terms of the distance travelled. For environments differing in noise levels, competitive pressures, and availability of resources, different minimal algorithms are required to achieve the best adaptation.

This work is being continued on a number of fronts. An infra-red beacon and matching receptor have been designed and built to allow measurement of stimulation over a range of up to 6 metres. The strength of stimulation falls off with the expected inverse square law. The angle of acceptance of the receptor is variable. By mounting the receptor on a mobile robot, these minimal algorithms can be tested for effectiveness in the real world; this validation is seen as necessary. In parallel, we plan to use the genetic algorithm to search the limited parameter space of these models in simulation under various types of environmental conditions; this is expected to uncover better rules than the few sampled so far. The benefits of asymmetry will be explored in further simulations by allowing the sensor to be placed asymmetrically, as is typical in phototactic flagellates. Finally, the work will be extended to examine the collective behaviour of many minimal agents with the ability to sense one another as well as the target; an initial simulation study of agents with 360° sensors has already been undertaken (Holland and Melhuish, 1996).

References

Braitenberg V. (1984) **Vehicles: experiments in synthetic psychology**. MIT Press, Cambridge MA

Feinleib M.E. (1980) Photomotile responses in flagellates. In **Photoreception and sensory transduction in aneural organisms**. Eds. F. Lenci & G. Colombetti, pp 45-68, Plenum: New York and London

Fraenkel G.S. and Gunn D.L. (1940) **The orientation of animals**. Oxford University Press. 2nd Edition: Dover Publ. Inc. New York 1961

Holland O.E. (1996) Grey Walter: the pioneer of real artificial life. Proceedings of 5th International Workshop on Artificial Life, Nara, Japan

Holland O.E. and Melhuish C.R. (1996) Getting the most from the least: lessons for the nanoscale from minimal mobile agents. Proceedings of 5th International Workshop on Artificial Life, Nara, Japan

Koshland D.E. (1980) **Behavioural chemotaxis as a model behavioural system**. Raven Press, New York

Kühn A. (1919) **Der Orientierung der Tiere im Raum**. Gustav Fischer Verlag, Jena

Loeb J. (1913) Die Tropismen. Hdb. d. vergl. Physiol. Bd. IV, 451-511. Hrsg. H. Winterstein. Gustav Fischer, Jena

Marken R.S. and Powers W.T. (1989) Random-walk chemotaxis: trial and error as a control process. Behavioural Neuroscience, Vol. 103, No. 6, 1348-1355

Martin H. (1964) Zur Nahorientierung der Biene im Duftfeld zugleich ein Nachweis für die Osmo-Tropotaxis bei Insekten. Z. vergl. Physiol. **48**. 481-533

Schöne H. (1984) **Spatial Orientation - the spatial control of behaviour in animals and man**. Princeton University Press

Walter W.Grey (1950) An Imitation of Life, Scientific American, May, pp 42-45

Walter W.Grey (1953) **The Living Brain**. Duckworth, London

Categorization in a real-world agent using haptic exploration and active perception

Christian Scheier and Dimitrios Lambrinos

AI Lab, Computer Science Department, University of Zurich
Winterthurerstrasse 190, 8057 Zurich, Switzerland
E-mail: {scheier,lambri}@ifi.unizh.ch

Abstract

An agent in the real world has to be able to make distinctions between different types of objects, i.e. it must have the competence of categorization. In mobile agents categorization is hard to achieve because there is a large variation in proximal sensory stimulation originating from the same object. In this paper we extend previous work on adaptive categorization in autonomous agents. The main idea of our approach is to include the agent's own actions into the classification process. In the experiments presented in this paper an agent equipped with an active vision and an arm-gripper system has to collect certain types of objects. The agent learns about the objects by actively exploring them. This exploration results in visual and haptic information that is used for learning. In essence, categorization comes about via evolving reentrant connections between the haptic and the visual system. Results on the behavioral performance as well as the underlying internal dynamics are presented.

1 Introduction

The ability to make distinctions is fundamental to adaptive behavior. It is one of the basic competences of any agent in the real world. Animals need to discriminate between food and non-food, they must recognize the nest, and they have to differentiate between conspecifics and individuals of other species. A robot for collecting garbage has to be able to distinguish between garbage and non-garbage, the garbage truck from other trucks, other garbage robots from human agents, it has to be able to recognize the charging station, etc. This competence is called classification or categorization. Some aspects of this capacity can be innate or pre-defined such as the taste of particular foods or the recognition of conspecifics. Others, like the shape of the food items and their location, might vary greatly, depending on the kind of environment. While a garbage robot might have a predefined way of identifying the charging station or of recognizing the garbage it should collect, the location of the charging station, or the visual appearance of the garbage might vary from city to city. There are several ways to endow a robot with the capacity to categorize. One way is to predefine the categories by either incorporating a special sensor (e.g. for color) or by predefining configurations in the sensory space that correspond to various categories. While the former is easily doable, we do not gain many insights into the nature of intelligence. The latter turns out to be difficult, especially if we take into account that the agents move around and that one and the same object leads to a large variety of proximal stimulation of the sensors. As an example, consider the visual space of the robot that was used to conduct the experiments presented in this paper. It is equipped with a CCD camera with approximately 2^{18} photoreceptors where each photoreceptor has 2^8 states, i.e. there are $2^{2^{21}}$ different states . Clearly, the problem of reducing the many degrees of freedom of this space is a non-trivial one. It is extremely difficult to define meaningful a priori configurations within this space. This implies that categories should be acquired at run time, i.e. as the agent is moving around in the environment.

Traditionally, categorization has been treated as an information processing question: the sensors receive a particular input which is processed and mapped onto an internal representation (e.g. a category node). This view dominates most psychological models of categorization. In these models categorization usually consists of associating some binary input vector to a category node via supervised learning schemes (see e.g.[7]). Similarly, in computer vision systems categorization is seen as a problem of matching the visual input to a stored representation or model of objects using methods such as direct template matching, hierarchical template matching or transform and match (see e.g. [6] for an overview). The problem encountered is that typically an enormous number of highly different sensory patterns should map onto the same representation. Efforts to solve this problem have only been successful to a limited extend. Biolog-

ical systems, on the other hand, are extremely efficient at categorizing their environment at run time (see e.g. [5]). One key difference between information processing approaches and categorization in natural systems seems to be that the former view categorization as a process which happens on the (visual) input side only whereas the latter heavily includes information stemming from the organism's own movements. For recognizing a memorized visual stimulus, for instance, flies have to shift the actual image to the same location in the visual field where the image had been presented during the storage process ([2]). In other words, the incorporation of action is crucial for the recognition process.

Self-motion is even more important for the development of categories in young infants. In the latter half of the first year, infants begin to actively manipulate the objects that they explore visually using several explorative activities such as mouthing, fingering, rotating or banging behaviors (e.g. [10]). This manipulation has the important consequence that the infant may acquire tactile and kinesthetic (i.e. haptic) information about the object through active touch. Recent studies indicate that during the first semester, infants can discriminate haptically among objects and transfer information from one modality to another (see e.g. [10] for an overview). Thus, from the earliest age, hands and mouth can be used to pick up information about objects. In coordination with other perceptual systems, the hands and mouth contribute to the perception of objects in the environment. There are active cross-modal comparisons, i.e. infants actively relate what they experience of the object in the haptic and visual modalities ([14]). In sum, infants discover object categories through the cross-correlation of multimodal experiences. Moreover, they disover such categories by acting upon the objects and exploiting the resulting multimodal experiences of objects (i.e. exploration). In our previous work we have developed an approach to adaptive categorization in mobile robots that is based on these results ([8],[9],[12],[13]). The main idea is to view categorization as a *sensory-motor coordination* rather than an isolated perceptual (sub-)system. This is achieved by including the robot's own actions into the classification process. In these experiments we have demonstrated how these ideas can be used to enable an agent to reliably learn a classification of objects in a straightforward and simple way. In this paper we considerably extend this framework. First, the sensory-motor complexity of the robot is significantly increased. While the sensory-motor system in the previous experiments consisted of IR sensors and two wheels the robot used in this paper is additionally equipped with a CCD camera and an arm with a gripper mounted at the end. Instead of passively scanning the camera an artificial eye has been implemented which actively moves a fovea to interesting parts (e.g. bright spots, texture, movement)

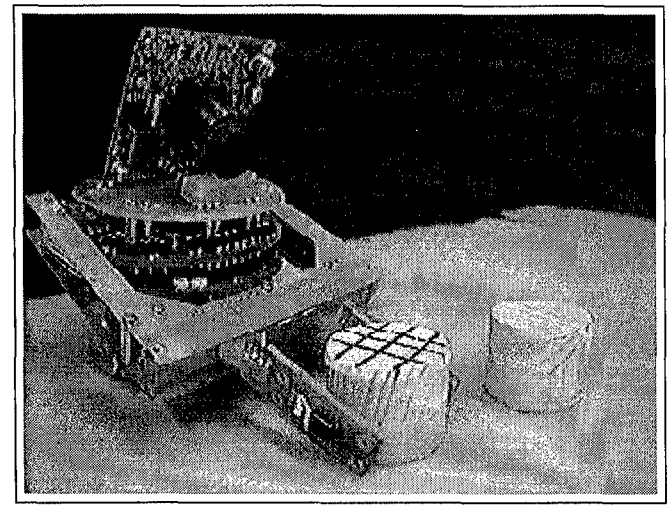


Figure 1: The robot and its environment. Explanations see text.

of the image. Once it has visually focused an object the robot starts exploring it using the arm-gripper system. Similar to what has been observed in infants, the robot uses the resulting multimodal (i.e. haptic and visual) experiences to learn about important properties (e.g. texture, conductivity) of the explored object.

A second improvement concerns the categorization mechanisms used. Previously, categorization was based on (a) learning a sensory-motor mapping using a temporal Kohonen map ([8],[12]) or growing dynamical cell structures ([9], [11]) and (b) associating this mapping with behavioral processes such as grasping, pushing or avoiding. The basic categorization mechanism was a conditioned association between the learned sensory-motor mappings and some behaviors. In the experiments presented in this paper categorization is not achieved via conditioning but rather by a learned *reentrant mapping* between visual and haptic feature maps. The term *reentry* refers to the fact that there are reciprocal connections between the feature maps. Reentry is necessary to account for the coordination of responses across modalities. When the robot explores an object the experience is visual, but also invokes haptic stimulation such as hardness or roughness of the object. Reentry is a mechanism to correlate these perceptual informations on the basis of their temporal contiguity ([4]). Thus, categories develop from the real-time correlations that exist accross the independent stimuli. This *correlative* function of reentry has been suggested to be one of the core mechanisms for the development of categories in infants ([14], see also [3]). A final extension of our previous work is the concept of *attentional sensory-motor loops* which are modulated by the category-specific responses of the reentrantly connected feature maps. In essence, categorization consists of breaking or enhancing the attentional sensory-motor loops depending on the type of object encountered and

the resulting activity in the feature maps. Each of these mechanisms will be explained in more detail below. The *task* of the robot is to collect certain types of objects. Since there are other objects in the environment which it should not collect, it has to learn (a) to recognize objects it should collect and (b) to discriminate the latter from all other objects in the environment.
The paper is structured as follows. First, the experimental set-up is described. Second, an overview of the architecture as well as a detailed description of the visual and the haptic system are given. Third, the results of several experiments are presented. Finally, the main problems and future work are pointed out.

2 Experimental Set-up

The mobile robot used in the experiments is is a $Khepera^{TM}$, 55mm in diameter and 32mm high (weight 70g) (see figure 1). The effector system consists of two wheels which are individually driven by DC motors, and an arm with a gripper installed at the end. The maximum object size that can be grasped with the gripper is about 40mm. There are two types of objects in the environment (see figures 1 and 2). Wooden objects with texture on the surface and others without texture. Textured objects have been made conductive by wrapping a metalic wire around them. The sensory system consists of a visual and a haptic system.

2.1 Visual system

Input to the visual system is provided by a miniature (1/3", 18g) monochrome CCD camera (ces VPC-465). The camera is equipped with a built-in lense with 90 degrees viewing angle. In order to reduce computational load all visual processing was done on a visual server based on a Pentium 133Mhz PC. The video image is sampled at a maximum of 30 fps using a video framegrabber (PMS from Media Vision). Images were grabbed at resolution of 640 x 480 pixels and were spatially averaged to provide an image of 160 x 120 pixels (see figure 2). The main part of the control architecture was run on a workstation (SUN SPARC 10). Communication between the visual server and the workstation was done using the TCP/IP communication protocol.

2.2 Haptic system

The input to the haptic system is as follows. Two wheel encoders provide position information with a resolution of 0.08mm. Arm and gripper position are sensed by position sensors coupled with the respective motors. The arm position sensor takes values from 0 (bottom back) and 255 (bottom forward), the gripper positions sensor takes values from 0 (open) to 255 (closed). Conductivity of objects can be read by a conductivity sensor which takes values from 0 (nonconductive, e.g. plastic, wood) to 255 (conductive, e.g. metal). Objects inside the gripper can be detected by an optical barrier that is mounted on the gripper. The optical barrier takes values from 0 (no object) to 255 (object presence). Finally, there are eight IR sensors, six in the front and two in the back with an angular resolution of 60 degrees each. The maximal distance which the IR sensors can detect is around 40mm. Because of the small distance that they can sense and their arrangement around the body surface of the robot we use them as skin (pressure) sensors.

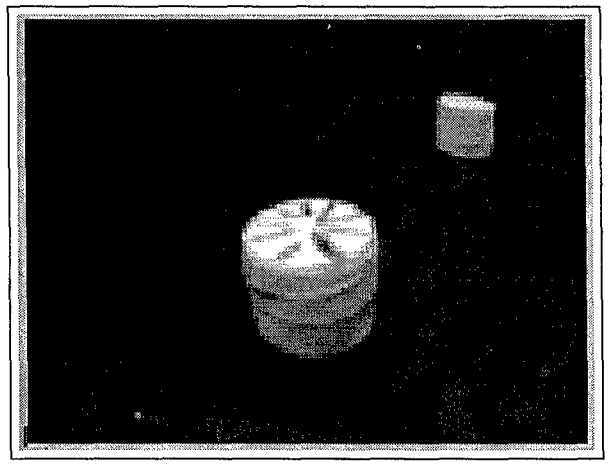


Figure 2: Original image (160x120 pixels) when the robot is in the front of a textured object with a non-textured object in the background.

3 Architecture

3.1 Overview

We first give an overall review of the complete system consisting of a visual and a haptic system linked via reentrant modifiable weights. An overview of the architecture is shown in figure 3. The present model consists of a total of about 2000 units and 140000 connections. A more detailed description of each individual system is presented below. The main ideas are as follows. We exploit the fact that the agent can interact with its environment in two ways: Firstly, there are focus of attention mechanisms which are directly coupled to the arm, gripper and wheel motors (see below). As a result the agent moves its body and the arm in a position where it can explore the object. Secondly, instead of categorizing visual information only, both haptic data which result from the exploration process as well as the focused visual data are used for learning and categorizing objects. As shown in figure 3, in both systems there are sensory maps that are connected to feature maps. Feature maps respond to properties of objects such as texture or conductivity. The interaction between the two modalities is implemented

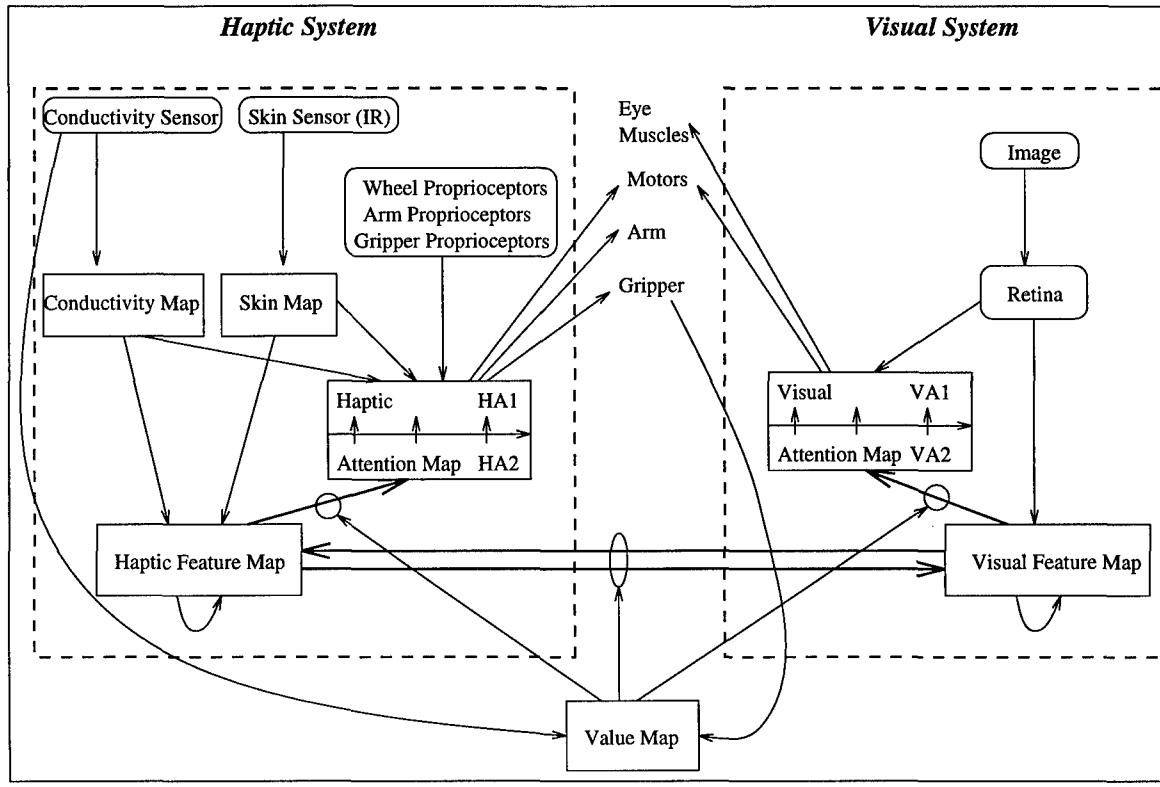


Figure 3: Overview of the architecture. There is a visual and a haptic system. Each system consists of sensory sheets, an attentional map and a feature map. There is a learned crossmodal interaction via reentrant connections between the haptic and visual feature map.

via modifiable reentrant weights between these feature maps. The correlation of signals of the haptic and the visual feature maps by these reentrant connections forms the basic mechanism of categorization. This is in accordance with the model of categorization in infants proposed by [14]. In sum, the reentrantly connected feature maps form a classification couple ([3]).

A fundamental property of our model is that the feature maps are connected via modifiable feedback connections to the attention maps (see figure 4). The main idea is to link the categorical responses of the classification couple to the attentional sensory-motor loop. In essence, the result of learning is that relevant objects enhance activity in the attentional loop while it is broken in the case of uninteresting objects (which in turn leads to ignoring of the object). Thus, there is no explicit avoidance or approach behavior linked to the classification couple. Instead, the dynamics of the classification couple lead to a differential modulation of the attentional sensory-motor loop depending on the encountered object.

3.2 The visual system

The visual system consists of a retina, a visual attention map and a visual feature map. We use an image of 160x120 pixels as an input to the *retina.*

An artifical fovea (20x20 pixels) is extracted from the center of the retina (see figure 5). The spatially averaged periphery of the retina gives input to area VA1 of the *visual attention map.* We have implemented several "attention operators" such as attention to texture, motion or bright spots. In the experiments presented in this paper only the latter operator was used. The visual attention map is part of an attentional sensory-motor loop in two ways. First, it is connected to the wheel motor map. These connections implement a continuous mapping between location of activity in the attention map and the resulting translational and rotational movements. For example, a bright object on the top left of the image will lead to high activation on the corresponding area in the attentional map and this in turn will be translated into forward translation and left rotation. Put shortly, the robot always orients its body towards areas of interest (bright spots in the present model). In addition, the attention map is connected to artificial eye muscles that move the fovea to bright spots. This saccading leads to a focussing of the fovea on interesting regions in the visual field. Thus, together with the motor map and the eye muscles the attention map forms a complete (visual) attentional sensory-motor loop: it brings the robot to relevant places (as defined by the attention operator) in the environment while at the same time keeping the eye

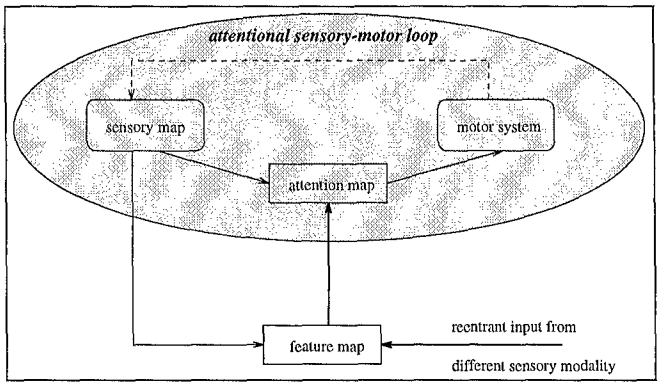


Figure 4: The feedback loop linking categorical responses to the attention system. There is a closed attentional loop between the sensory map, the attention map and the motor system. The resulting sensory-motor coordination is modulated via the modifiable connections between the feature map and the attention map. The main result is that categorical responses are linked in a continuous way to the attentional loop: Relevant objects will enhance activity in the attention map and thus strengthen the sensory-motor coordination. On the other hand, the encounter of uninteresting objects will result in low activations in the attention map and consequently break the attentional loop (sensory-motor coordination).

Field	Haptic	Visual
Feature Map	100	100
Attention Map	200	600
Sensory Maps	80(Skin), 100(Conductivity)	700(Retina), 400 (Fovea), 300 (Periphery)
Proprioceptors /Motor Maps	10(Wheels), 10(Arm), 10(Gripper)	
Value Map	20	

Table 1: The number of units per region on both haptic and visual system.

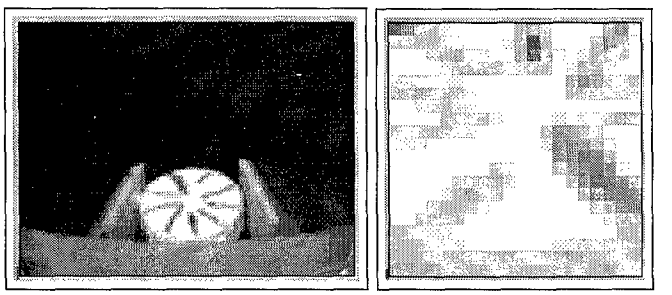


Figure 5: Original image when the robot explores a textured object and the corresponding foveal activity. In this case the foveated region is approximately in the center of the object.

focussed on the spot where the robot is heading to.
The fovea in turn is connected to the *visual feature map*. The feature map responds to relevant features in the fovea. In the present model these are defined to be horizontal and vertical edges, i.e. texture (note that from a distance the cylindrical objects used in the experiments appear as vertical edges; this is not a problem, however, since learning occurs only when the robot has explored an object, i.e. when it is close the the object). The visual feature map contains excitatory and inhibitory units; excitatory units make connections to neighboring excitatory units, thus enhancing activity locally. Inhibitory units influence excitatory units further away, thus providing lateral inhibition. This connectivity scheme leads to local and constrained centers of activations. The visual feature map is connected back to area VA2 of the visual attention map via modifiable weights. The main idea behind these weights is that the categorical responses of the classification couple (i.e. the two feature maps) should modulate the attentional sensory-motor loops by either enhancing or breaking it. Area VA2 consists of a population of inhibitory and excitatory units that are connected via prewired weights to area VA1. The activity of these units increases as the weights from the feature map to the attention map evolve. These weights are topographic in the sense that areas that code for strongly textured input are mapped onto the excitatory population, while areas that code for bright or non-textured objects are mapped onto the inhibitory population. High activity of the inhibitory population will lead to a suppression of activity in VA1 and in turn to a breaking of the visual attentional loop. The value map implements a general bias or motivation in the system. The value map receives input from the resistivity sensor and the gripper proprioceptors. The basic motivation behind these connections is that the robot should learn only when it explores an object. Activity in this map acts like a gating function and is used as a reinforcement signal for the synaptic modifications between the feature and attentional map as well as for the reentrant connections between the two feature maps. Finally, the visual feature map is connected to the haptic feature map via reentrant connections. Again, the synaptic growth of these connections is modulated by activity of the value map. The value map is strongly active when the robot has something in the gripper. In this way the robot only learns about objects when it is exploring them with its arm-gripper system. The dynamics of the classification couple, the synaptic growth between the classification couple as well as between the feature maps and the attentional maps will be illustrated in the results section below.

3.3 *The haptic system*

The overall functionality of the haptic system is similar to the one just described for the visual system. Again, there are sensory maps, an attention map and a feature map. The sensory maps consist of a *conductivity map*, and a *skin map*. They get input from a conductivity and a skin sensor, respectively. Both maps are connected to area HA1 of the *haptic attention map*. Similar to the visual equivalent, the haptic attention map and the wheel, arm and gripper motor maps form an attentional sensory-motor loop. Again there is a continuous mapping of location of activity in the attentional map and the resulting translational and rotational movements. The main result is that as soon as the robot has made body contact with an object (which is sensed by the skin sensors) it will bring the object to the front of its body. This "haptic focussing" leads to an increase of activity in area HA1 of the attention map and in turn to large activity in the arm motor map, causing the robot to lower it's arm. Lowering the arm leads to increased activity in the arm proprioceptors and in turn to an even more increased activity in HA1. As a result of this the robot will start exploring the object by closing the gripper. Area HA2 consists of inhibitory and excitatory populations of units. They receive inputs from the haptic feature map map and they are fully connected with the units of area HA1. The projections between the haptic feature map and HA2 are topographic in the sense that areas that code for non-conductivity are maped onto the inhibitory population, while areas that code for conductivity are mapped onto the excitatory population. HA2 and HA1 are fully inteconnected. High activity of the inhibitory population will lead to a suppression of activity in HA1. As a result the gripper will be opened and the arm will be lifted. At the beginning of a trial the main source of activity in the haptic attention map stems from the skin map. This is a kind of "haptic reflex" that makes the robot haptically track and explore objects. Over time connections between the haptic feature map and the haptic attention map (area HA2) evolve. This leads to an amplification of attention for relevant objects and to a breaking of the haptic attentional sensory-motor loop when the robot encounters irrelevant objects.

3.4 *Activation and learning rules of neuronal fields*

The generic equation that describes the activation rule for the attention maps is:

$$a_i^{AM}(t) = \sum_{i=1}^{N^{FM}} a_j^{FM}(t) w_{ij}^{FM}(t) + \sum_{i=1}^{N^s} a_j^s(t) w_{ij}^s(t) \quad (1)$$

where $a_i^{AM}(t)$ is the i-th unit of the attention map, $a_j^{FM}(t)$ is the activation of unit j of the feature map, $w_{ij}^{FM}(t)$ is the weight from unit j in the feature map, $a_j^s(t)$ is the activation of unit j of the sensory map and $w_{ij}^s(t)$ is the weight connecting unit$j$ to unit i. In the experiments presented below, updates are made approximately every 100 milliseconds. The activity of the units in the feature maps, $a_i^{FM}(t)$, is computed using a "leaky integrator" activation function [1]:

$$a_i^{FM}(t) = \beta a_i^{FM}(t-1) + \alpha[\sum_{i=1}^{N^r} a_j^r(t) w_{ij}^r(t) + \sum_{i=1}^{N^s} a_j^s(t) w_{ij}^s(t)] \quad (2)$$

where $0 < \beta < 1$ relates to the time constant of the unit, $0 < \alpha < 1$ is the attack parameter, $a_j^r(t)$ is the activation of unit j of the other reentrantly connected feature map, $w_{ij}^r(t)$ denotes the reentrant connection strength from unit j to unit i, $a_j^s(t)$ is the activation of unit j of the sensory map,$w_{ij}^s(t)$ is the weight connecting unit j to unit i ($\alpha = 0.8$, $\beta = 0.2$). The connections between the sensory map and the feature map are chosen such that they result in a population coding of the average sensory activity (conductivity/skin map activations for the haptic and texture for the visual feature map, respectively). Equation 2 makes the activation of the units not only depend on the weights and the current inputs but also on the activation of the previous time step. Besides being a biologically plausible activation function, the leaky integrator function has the important property that it leads to stability of responses and a kind of "low pass" filtering on the input. The connection strenghts between the haptic and the visual feature map, and between the feature maps and the attention maps are computed using a Hebbian learning scheme:

$$\Delta w_{ij} = v(t)(\eta a_i(t) a_j(t) - \epsilon a_i(t) w_{ij}(t)) \quad (3)$$

where w_{ij} represents the strength of the connection between the presynaptic unit j and the postsynaptic unit i, $v(t)$ is the value signal (activity of the value map), $a_j(t)$ is the activation of the pre-synaptic unit, and η, ϵ are the learning rate and decay parameters respectively ($\eta = 0.02$,$\epsilon = 0.1$).

4 Results

Experiments were conducted on a flat arena (100 cm x 100 cm) with walls (8 cm height) on each side. Objects were of 1.5 cm in diameter and 2 cm high, the shape of the objects was cylindrical (see figure 1). There were conductive and non-conductive objects in the environment. The conductive objects had a strongly textured surface while the non-conductive had a white or only slightly textured surface. The robot's task was to collect the conductive objects. We present results on several levels. First, the overall performance of the system is quantified on the behavioral level. Second, the categorization dynamics are illustrated by measuring the

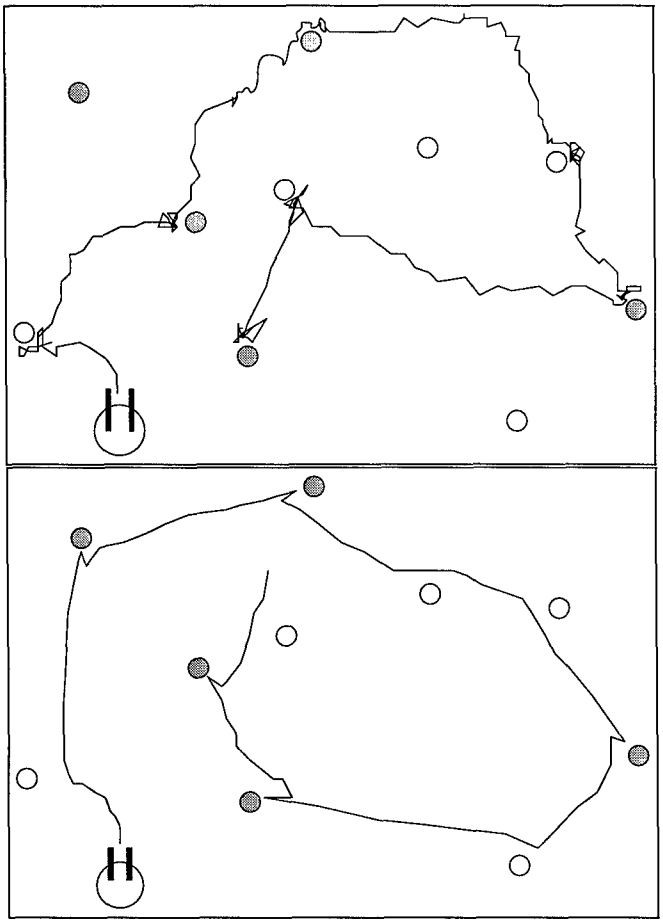


Figure 6: A typical trajectory of the robot before and after learning had occured. For visualisation purposes objects were removed from the gripper once the robot had grasped them.

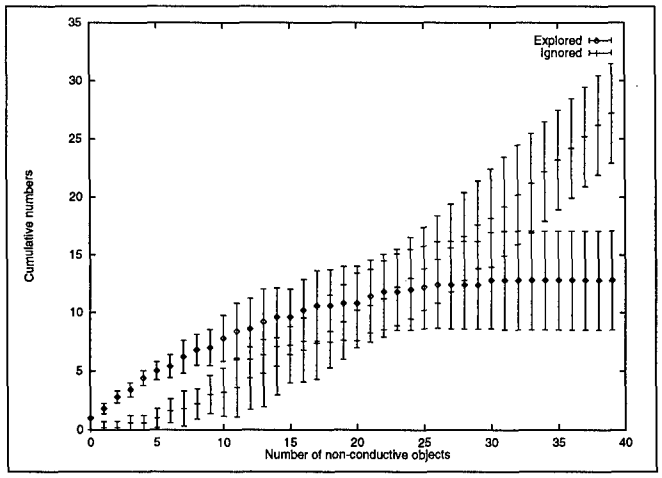


Figure 7: Cumulative number of exploration and ingoring steps for 40 non-conductive objects. The data are means ($\pm$ sdev) over 20 trials.

synchronization of activity in the classification couple. Third, the performance and the categorization dynamics are traced back to the dynamics of synaptic change. The behavior of the robot as it moves around in the environment and explores objects is shown in figure 6. The trajectories were recorded with a video camera and then hand traced. Figure 6 (*top*) shows a typical trajectory at the beginning of a trial. White and shaded circles indicate non-resistive/non-textured and resistive/textured objects, respectively. It can be seen there is no distinct behavior for the two types of objects. Rather the robot approaches all objects and explores them. Figure 6 (*bottom*) shows a typical trajectory after the robot has encountered 10 objects of each type. Two main results can be taken from the traces in figure 6. First, the robot has stopped exploring both types of objects. Rather the behavior is now governed by the dynamics of the classification couple. Second, the robot "ignores" non-conductive objects while it grasps the conductive ones (without first exploring them). We use the term "ignoring" instead of "avoidance" to indicate that there is no seperate avoidance module. Rather the avoiding is achieved by breaking the attentional sensory-motor loop. In order to quantify this learning process the number of non-conductive objects explored and ignored was recorded for 20 trials. Each trial ended when the robot had encountered 40 non-conductive objects. Figure 7 shows the averaged cumulative number of non-conductive objects the robot explores and ignores.

At the beginning of the trials the agent always explored non-conductive objects. After it had explored around 6 objects it started to ignore them. Because the weights had not been sufficiently evolved it still explored some of the objects. After having encountered around 12 objects the robot only explored because of errors in the sensory readings. Since the main interest of this paper is on categorization, the dynamics of the classification couple during a typical trial was recorded. One of the main consequences of using reentry as the mechanism for categorization is that temporal correlations or synchronizations in the activity between neuronal areas emerge. In order to quantify these correlations we estimated the level of synchronization between the visual and the haptic feature map. We have quantified synchronization as the product of the average activities of the areas V1, V9 and H1, H9, in the visual and haptic feature maps, respectively. Units in area V1 selectively respond to non-textured regions in the fovea while units in V9 respond to textured ones. Similarly, H1 and H9 of the haptic feature map selectively respond to non-conductive and conductive stimuli, respectively. Figure 8 shows the evolution of synchronization between the classification couple as the robot explores objects. Three main results can be taken from this figure. First, the *level of synchronization* between areas V1, V9 and areas H1, H9 (denoted

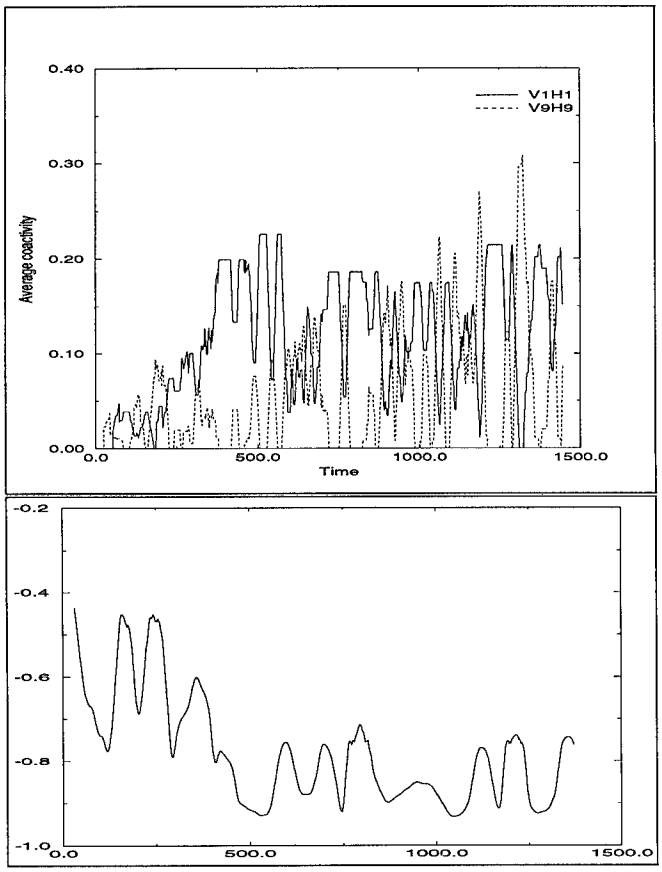


Figure 8: *Top.* Level of synchronization between areas H1, V1 and areas H9, V9. *Bottom.* Correlation of level of synchronization between H1V1 and H9V9. See text for explanation.

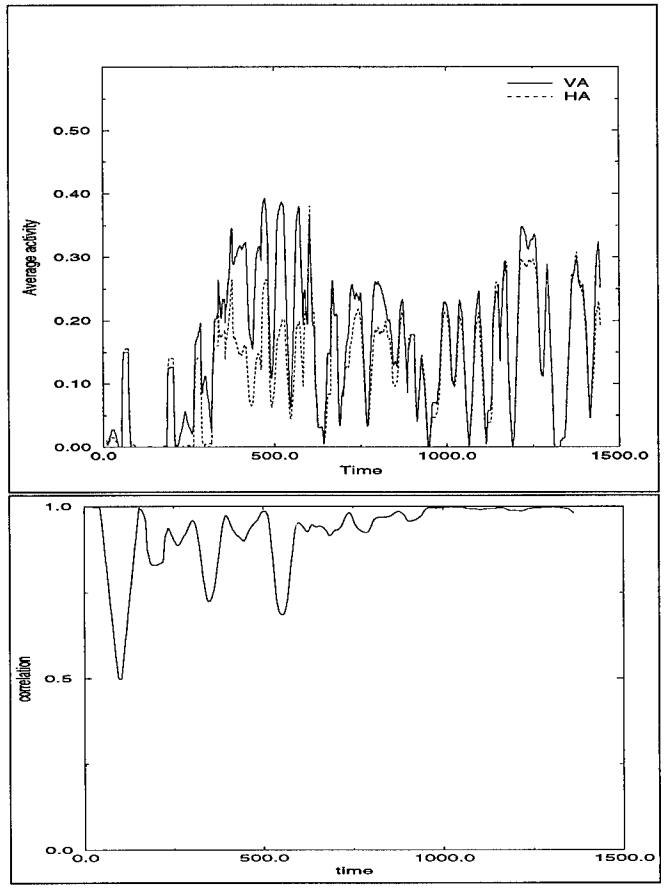


Figure 9: *Top.* Level of synchronization between areas HA2 and VA2 of the haptic and the visual attention map, respectively. *Bottom* Correlation between level of synchronization of HA2 and VA2. See text for explanation.

as V1H1 and V9H9 in figure 8, *top*) at the beginning of the trial is significantly lower than after learning has occured. For example, until around 300 steps the mean synchronization level is around 0.05 while later in the trial it increases up to 0.3 implying levels of average activity around 0.55. This increase is due to the evolving reentrant weights that couple the activities in the two feature maps (see below). Second, the *difference* in the synchronized activities between the two areas (V1H1 and V9H9) increases significantly. This can also be seen from the negative correlations that evolve between the coactivities of V1H1 and V9H9 (see figure 8, *bottom*). Third, the *coherence* of synchroneous activity between the maps increases. As can be seen in the figure, the dynamics of the coactivities become strongly coupled over time. Again, this is reflected in the correlations V1H1-V9H9 that reach an average level of around -0.9 (see figure 8, *bottom*).

Another important aspect of our model is that the categorical responses (i.e. the synchronised activities) in the classification couple modulate via modifiable weights the dynamics of the attentional fields. Figure 9 illustrates these couplings for the same trial as the data in figure 8. It depicts the synchronization of the average activity of the inhibitory populations in areas HA2 and VA2 of the haptic and the visual attention map, respectively. Note that although these two areas are not directly connected (see figure 3) a strong synchronization emerges between between the two areas starting at around 300 steps and reaching a maximum correlation of 1.00 at around 1000 steps (see figure 3 *bottom*). This is due to the evolving weights between the feature maps and the attentional maps which leads to a projection of the coupled feature map activities onto the attentional maps. Similar dynamics emerge in the excitatory populations of the attention maps (data not shown).

The activation patterns just described can be traced back to the dynamics of synaptic change between the respective areas. Figure 10 shows the development of synaptic strength between areas V1, V9 of the visual feature map and areas H1, H9 of the haptic feature map. Again, V1 and V9 denote units tuned to non-textured and textured regions in the fovea, respectively, while H1 and

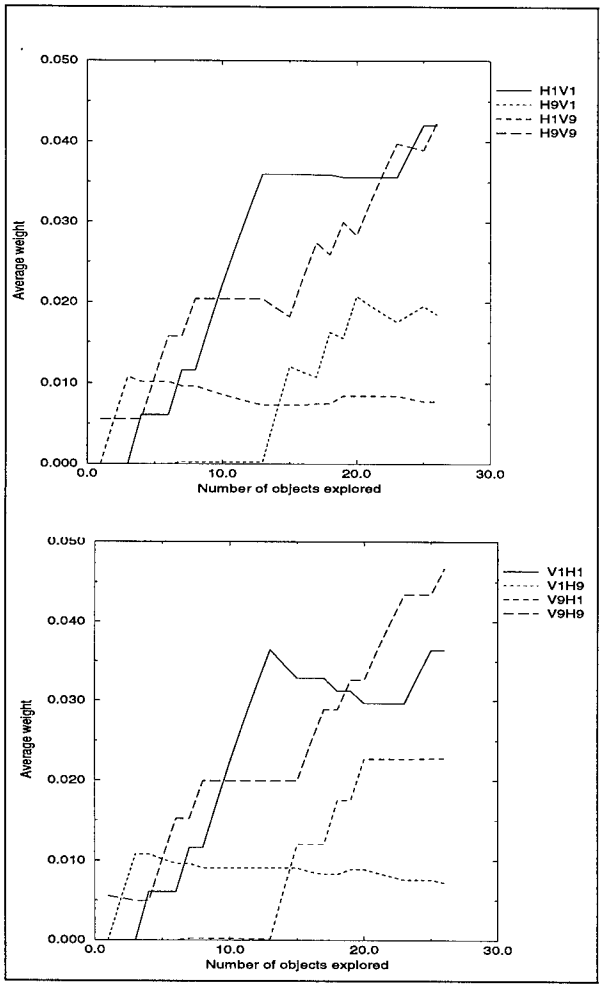


Figure 10: *Top* Average weights from the haptic to the visual feature map and from the visual to the haptic feature map (*Bottom*) for areas V1, V9, H1 and H9 as a function of the number of objects explored for a typical run. See text for explanation.

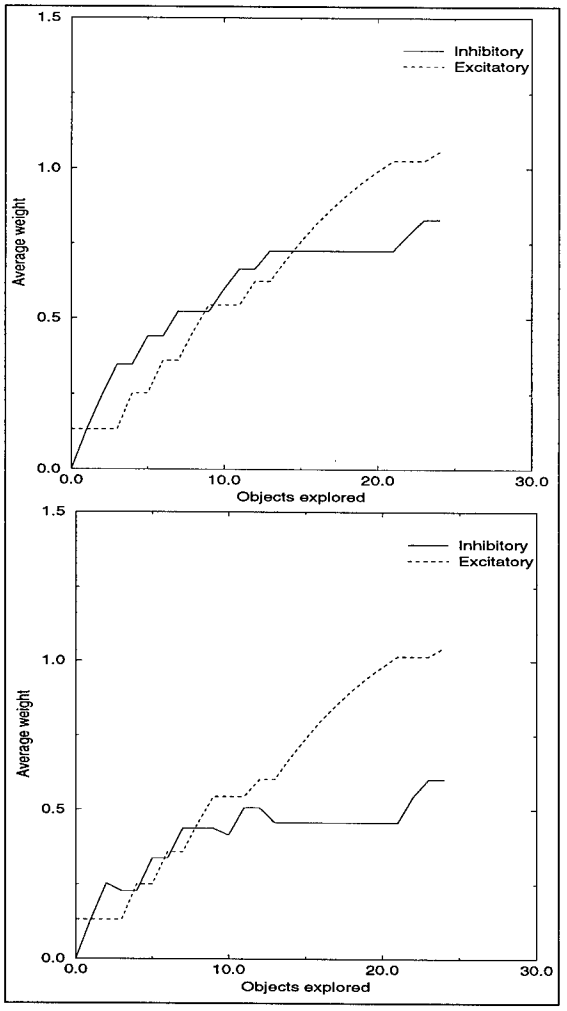


Figure 11: *Top* Average weights from the haptic feature map areas to the haptic attention map and from the visual feature map to the visual attention map (*Bottom*) for excitatory and inhibitory populations for a typical run. See text for explanation.

H9 are regions in the haptic feature map tuned to non-conductive and conductive stimuli, respectively. The synaptic strengths between the two feature maps show a similar pattern. Weights between areas responding to stimuli which are reinforced by the value map, i.e. weights between areas V1H1, V9H9, H1V1 and H9V9, increase faster and reach a higher level than the ones which are of no value to the agent. The latter connections (V1H9, V9H1, H1V9 and H9V1) stem from activity in the feature maps which are mainly caused by errors in the sensory maps (i.e. errors in the sensor data). Note that these erroneous weights decrease over time due to the decay term in equation 3. The steepness of the curves in figure 10 is a function of the sequence of objects the agent encounters. For example, it can be seen in figure 10 that after having explored around eight objects the agent encountered three non-textured and non-resistive objects. As a result, the weights H1V1 and V1H1 show a significant increase during that period.

Finally, figure 11 shows the evolution of the average weights from the feature maps to the attentional maps. These weights couple the categorical responses to the attentional sensory-motor loops. As can be seen in the figure, weights to both excitatory and inhibitory populations in the attention maps increase significantly with the number of objects the agent has explored. More specifically, area V1 of the visual feature map (responding to non-textured stimuli) becomes associated with the inhibitory population of the attention map (area VA2, see figure 3). Strong weights also evolve between area V9 (responding to textured stimuli) and the excitatory population of the visual attention map. Together these connections lead to a suppression of activity in the visual attention map for non-textured objects while the encounter of textured objects leads to an enhanced activity. Sim-

ilar weights evolve between the haptic feature map and the haptic attention map. Area H1 (coding non-resistive input) becomes associated with the inhibitory population of the haptic attention map while area H9 which selectively responds to conductive objects becomes associated with the excitatory population of the attention map. Again, these connections have the consequence that the activity in the haptic attention map gets supressed for non-resistive objects and enhanced for resisitve ones.

5 Discussion

In this paper we have addressed the problem of categorization in autonomous agents. We have presented an architecture that is based on the concept of sensory-motor coordination rather than the one of information-processing. Sensory-motor coordination was used on different levels. First, a saccading mechanism was used to focus the fovea on interestings regions in the environment. The sensory-motor coordination here is a reflex based on the fovea and the eye muscles. It leads to a significant decrease of the number of degrees of freedom in the visual space because only the foveated part of the image has to been considered. Second, so-called attentional sensory-motor loops were used. The visual attentional loop caused the robot to orient and move towards objects while the haptic attentional loop resulted in a more fine-tuned focussing of the object with respect to the body of the robot. As a result the robot could explore the object by lowering the arm over the object and closing its gripper. Finally, sensory-motor coordination was involved in the category learning itself. This is because the agent only learned when it had haptically explored an object which in turn is the result of the visual and attentional sensory-motor loops. An important aspect of this is that the categorization is embedded in the interaction of the agent with its environment.
We have shown that based on this architecture the agent is able to learn to categorize the objects in its environment. The categories that evolve are expressed in the synchronization of activity in the classification couple.

References

[1] G. J. Chappell and J. G. Taylor. The temporal kohonen map. *Neural Networks*, 6:441–445, 1993.

[2] M. Dill, R. Wolf, and M. Heisenberg. Visual pattern recognition in drosophila involves retinotopic matching. *Nature*, (365):751–753, 1993.

[3] G. M. Edelman. *Neural Darwinism.* Basic Books, New York, 1987.

[4] O. Sporns G. Tononi and G.M. Edelman. Reentry and the problem of integrating multiple cortical areas: simulation of dynamic integration in the visual system. *Cerebral Cortex*, 2:310–335, 1992.

[5] R. J. Herrnstein. Levels of categorization. In W. E. Gall G. M. Edelman and W. M. Cowan, editors, *Signal and Sense. Local and Global Order in Perceptual Maps*, pages 365–413. Wiley, New York, 1993.

[6] J. E. Hummel. Object recognition. In M. A. Arbib, editor, *The hand book of brain theory and neural networks*, pages 658–660. MIT Press, Cambridge, MA., 1995.

[7] J. K. Kruschke and M. A. Erickson. Five principles for models of category learning. In Z. Dienes, editor, *Connectionism and Human Learning*, pages 365–413. Oxford University Press, Oxford, 1995.

[8] D. Lambrinos, C. Scheier, and R. Pfeifer. Unsupervised classification of sensory-motor states in a real world artifact using a temporal kohonen map. In *Proceedings of the International Conference in Artificial Neural Networks ICANN'95*, pages 467–472, Paris, October 1995.

[9] R. Pfeifer and C. Scheier. Sensory-motor coordination: the metaphor and beyond. *Journal of Robotics and Autonomous Systems*, in press.

[10] P. Rochat. Object manipulation and exploration in 2- to 5-month-old infants. *Developmental Psychology*, 25:871–884, 1989.

[11] C. Scheier. Incremental category learning in a real-world artifact using growing dynamical cell structures. In M. Verleysen, editor, *Proceedings of the European Symposium on Artificial Neural Networks ESANN96*, pages 656–667, Brussels, Belgium, 1996. D facto Publications.

[12] C. Scheier and D. Lambrinos. Adaptive classification in autonomous agents. In R. Trappl, editor, *Cybernetics and Systems Research*, pages 1037–1043, Vienna, Austria, 1996.

[13] C. Scheier and R. Pfeifer. Classification as sensory-motor coordination: a case study on autonomous agents. In *Advances in Artificial Life. Proceedings of the Third European Conference on Artificial Life ECAL95*, pages 656–667, Granada, Spain, June 1995.

[14] E. Thelen and L. B. Smith. *A Dynamic Systems Approach to the Development of Cognition and Action.* Bradford Books, Cambridge, Massachusetts, 1994.

How to attract females: further robotic experiments in cricket phonotaxis

Barbara Webb &
AI Group
Dept. of Psychology
University of Nottingham
Nottingham NG7 2RD
U.K.
email:bhw@psyc.nott.ac.uk

John Hallam
Dept. of AI
University of Edinburgh
5 Forrest Hill
Edinburgh EH1 2QL
U.K.
email: john@aifh.ed.ac.uk

Abstract

Choice behaviour - preferentially approaching one out of several conspecific male calling songs - is considered an advanced feature of the female cricket's auditory localization ability, and is often studied from an evolutionary perspective. However it is not clear that this behaviour is not simply a consequence of the interaction of the particular localization mechanism of the cricket with the environmental constraints of spatially separated sound sources. In this paper we approach this issue through a 'neuroethological' analysis of the behaviour of a simple robot model of the cricket. By correlating internal and external measurements during behaviour it is shown that simple choice between two signals can occur without explicit recognition mechanisms, and performance is not degraded relative to a single source despite a noisier sensory situation. Plans to improve the model and extend the tests, to take into account more recent biological data, are described.

1. Introduction

By linking robotic and biological investigations of sensorimotor control, the same problem can be studied through both analysis and synthesis, gaining the advantages of each. By studying a particular biological system we can specify in detail the behavioural competence we would like to emulate, and discover mechanisms, tuned by millions of years of evolution, by which it can be acheived. By building a particular robotic system we are forced to fill out complete hypotheses of how the biological mechanisms might work, and can directly test how well the hypotheses explain the behaviour under a comparable range of physical conditions.

These advantages were well illustrated in our previous work, building a robot model of the female cricket's ability to locate males by approaching their calling song (Webb, 1994). A successful and robust device for locating specific sound-sources was developed that utilised the unique properties of the cricket auditory system for directional hearing. In the process, a simple alternative for producing the recognition-like behaviour of the cricket was devised and could be shown to suffice to explain evidence that was previously interpreted as indicating the existence of dedicated neural circuitry for recognition in the cricket.

Another advantage of adopting a particular animal model is that it can provide a whole range of behaviours to emulate. For example we are now examining the cricket's escape behaviour and the neurophysiology of its polarisation vision, with the possibility of integrating multiple sensory systems on one robot model. Here, we discuss the investigation of a further behavioural, rather than sensory, competence: the choice-like behaviour of female crickets when faced with multiple male calling songs. Does this behaviour require neural circuitry for identifying, comparing and choosing between different songs? Or might the choice in this situation be like the recognition - reproducible by the interaction of a certain kind of localization mechanism with the environmental constraints?

In this paper we approach these questions through a 'neuroethological' analysis of robot behaviour: that is, making simultaneous recordings of the internal processing and external performance to discover how the mechanism actually functions in real situations. This extension of the robot modelling methodology yields some interesting insights into the interpretation of neurophysiological findings in general, and into the cricket system in particular. Further investigations will incorporate new evidence about the cricket, derived from the previous robot work and recent biological experiments, utilising physical model sensors that can be dynamically adjusted to more precisely tune the behaviour of the robot to the animal system.

2. Cricket choice behaviour

Cricket males produce calling songs of a characteristic frequency and temporal pattern (rapidly repeated pure tone bursts) and females must display at least sufficient selectivity to approach conspecific males rather than any other sound source. It has been suggested that they make further selection amongst conspecific males, in which certain song parameter values (such as faster repetition rates) are preferred (Popov & Shuvalov, 1977; Simmons, 1988; Hedrick & Dill, 1993; Wagner et. al. 1995).

Examples such as this of female mate choice are of particular interest in evolutionary theory. On one hand, females may have evolved preferences that reflect the fitness of the singing male. On the other hand, an arbitrary preference by females may drive males to evolve to fit that preference even though it does not otherwise contribute to their fitness. If females prefer louder songs, for example, there will be a trade-off for the male between increased chances of mating and increased risk of predation. There are a number of theoretical models for such interactions (Anderson, 1994).

Attempts to show that male characteristics such as size are correlated with the song preferences of females have been inconclusive so far. Simmons (1988) found intensity balanced songs from larger males also attracted more females. But Bailey et al. (1990) suggest it may generally be the closer male rather than the larger that the female approaches. Weber and Thorson (1988) suggest that females almost always show a preference for whichever song is louder or contains more power. Stout and McGhee (1988) looked at variations along a number of parameters of the song and found some trade-off between amplitude and syllable repetition rate in the choice behaviour.

Wagner et al. (1995) suggest four possible explanations for cricket female preference for certain songs:

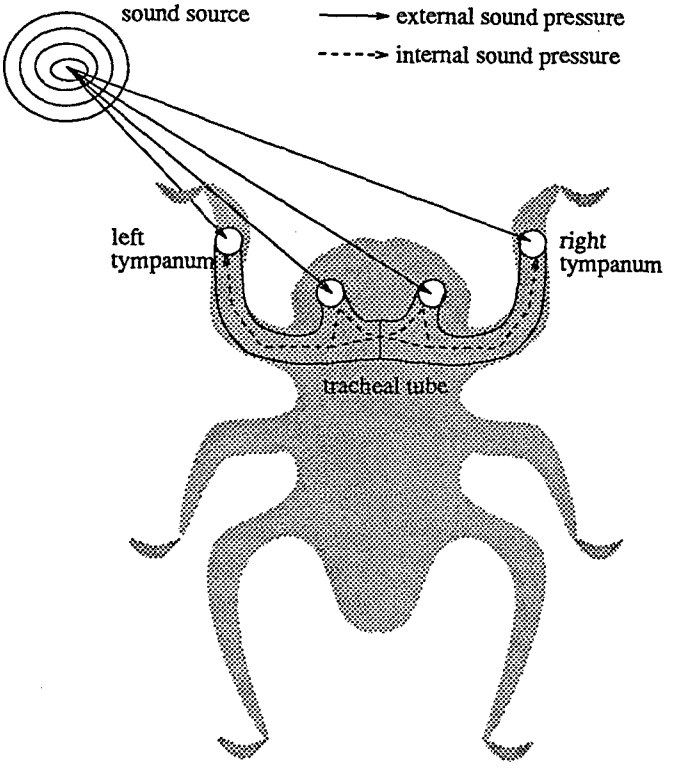
1. biases in the sensory or processing system that evolved in another context
2. immediate benefit from mating with males producing certain songs
3. the greater localizability of these song types
4. a heritable fitness benefit associated with these songs

Although these hypotheses are not mutually exclusive, each suggests rather different consequences for the actual mechanism that enables the female to show preferences. In particular, 1 & 3 suggest that the preference is a consequence of the design of the localization system and 2 & 4 suggest the design of the localization system is (at least partly) a consequence of the preference. It seems sensible to rule out the possibility that females localize preferentially only as a side effect of being able to localize at all, before invoking more truly decision-like processes in which "females evaluate potential mates on the basis of their calls" (Hedrick, 1995). It should also be asked at what stage during the processing of the signal the preference can be observed - in the peripheral ear response, the neural processing or only in the behavioural output?

3. Cricket & robot auditory processing

The cricket's auditory apparatus comprises an ear (tympanum) on each leg and auditory spiracle on each side of the body, all connected by internal tracheal tubes. The auditory signal transduced from the tympani is a result of the combination of the external soundfield incident on the auditory apparatus with the sound that propagates through the internal tracheal system to impinge on the rear walls of the tympani, which will be phase delayed by an amount proportional to the distance through the trachea and the direction of the sound (see figure 1). This system was mimicked in the robot using a simple phase delay circuit; for a particular frequency this gives a reliable direction-dependent difference in the response of the robot's virtual tympani, so the one nearer the sound source has a larger amplitude. It was argued that the frequency selectivity in the cricket's behaviour was largely a consequence of this system for directional hearing.

Figure 1: *The cricket auditory apparatus. The response amplitude at the tympanum depends on relative phase of the internal propagated sound pressure and external sound pressure, which varies with the direction of sound.*



Recent analysis by Michelson et al. (1994) of the cricket's peripheral auditory system's properties has added in practice detail to the previous in principle understanding of this pressure difference receiver. They show that the signals propagating from the ipsilateral and contralateral spiracles are significantly stronger than that from the contralateral tympanum, and they estimate the transfer functions that couple the spiracles to the tympani. This reveals that the frequency tuning is even more pronounced, as the precise phase delay for the cricket song wavelength is produced by special acoustic properties of the tracheal tube. We are currently constructing an electronic hardware emulation of this system that permits crucial parameters such as propagation delays and transfer gains to be programmed (see figure 2).

The response of the tympani or auditory neurons in complex, multisource, sound fields has not been well studied. In particular it is unclear whether the behavioural preference for louder sounds can be simply explained as a function of the auditory system response. That is, might the pressure difference receiver when stimulated by two sound sources of different amplitudes produce an output that is dominated by the louder and filters out the weaker signal? Or once the animal starts to move towards one sound source, is the effect of the other source on the auditory response minimal? If so, no higher processes would be needed to explain at least this aspect of 'choice' ability.

4. Cricket and robot neural processing

In the cricket, the tympanal response amplitude is coded neurally both in the firing rate and firing latency of auditory interneurons, which have low-pass filtering characteristics. In the robot, the latency difference was reproduced with a leaky integrator equation, and used to determine the turning direction - turn to the side that fires first (see figure 3). It was shown that this can suffice to explain the selectivity for conspecific temporally patterned songs, as song pattern affects the clarity and rate of latency cues.

Figure 2: Schematic of the new auditory circuit. Each auditory opening is modelled by a microphone and pre-amplifier. The four auditory sources are combined into two resultant signals, modelling the combination occurring at the tympani, by a set of programmable delays and weighted summations. The two resultant signals are then rectified to give outputs, proportional to their respective intensities, that model the excitation of the two tympani under the influence of the sound field. The circuit used for experiments in this paper used two sources, a fixed delay, and unweighted summation in software as the relevant parameters were then unknown.

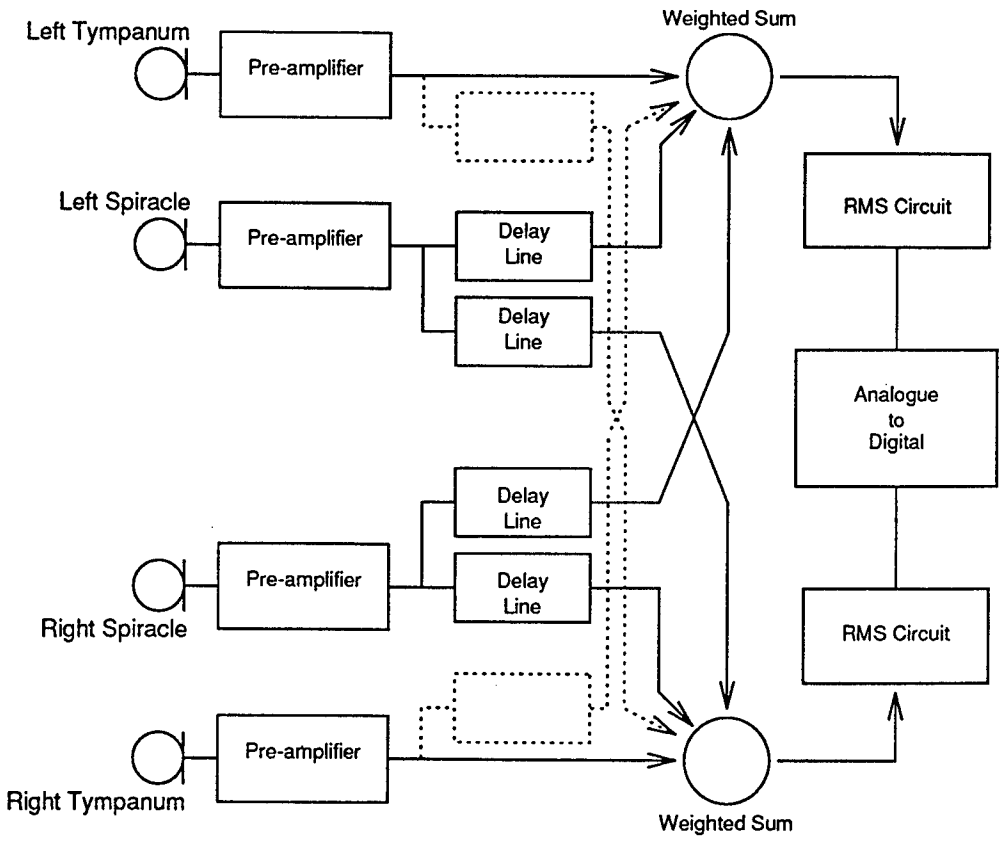
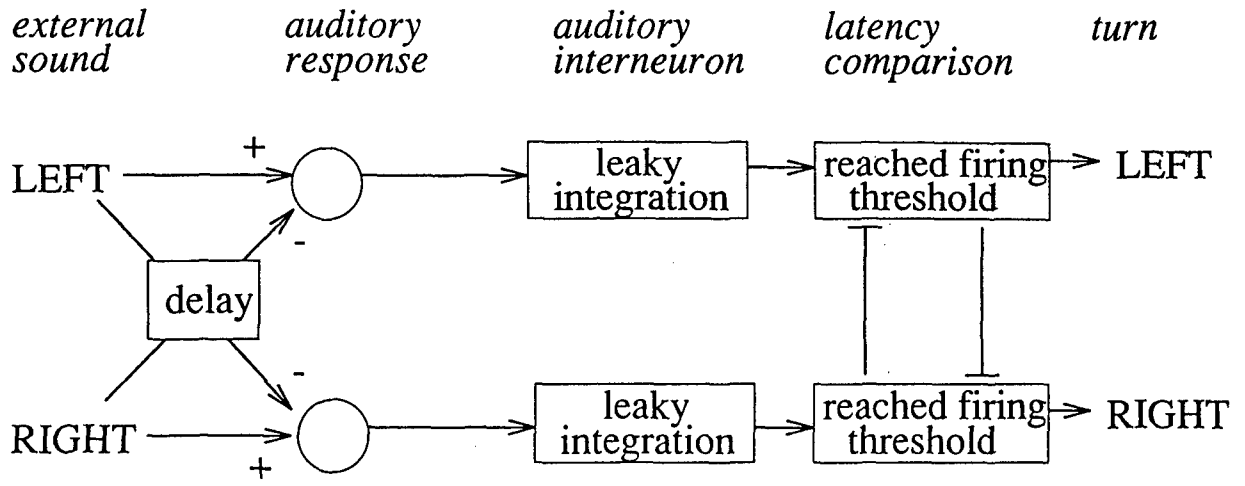


Figure 3: Proposed control mechanism for cricket phonotaxis, implemented in the robot. Phase delay circuit creates a direction dependent auditory response; leaky integration of that response by 'auditory interneuron' (AN) creates latency differences; the first side to reach threshold causes a turn to that side and suppresses response to the other side.



Many biologists believe that firing rate comparison is more likely to be the cue used by the cricket, especially because in recording from the interneurons firing rate appears to reflect far more clearly than latency the direction of the sound source. Nevertheless, one prediction from the robot model - that latency differences alone will suffice to cause turning where firing rate is equal - was recently confirmed independently in cricket experiments (von Helverson and Pollack, pers.comm.).

Again the response in the two-sound situation has not been well-studied. In particular, does the second sound differentially affect the clarity of firing rate vs. latency as a cue for turning? If the average activity of the auditory interneurons is substantially changed by multiple sounds, but the behaviour is not, this would be further support for the latency comparison model.

There are cricket experiments that seem to show such results: Stabel and Wendler (1988) found that with a continuous tone on one side producing a high firing rate, crickets turned to the other side where the song pattern could be heard; Pollack (1986) using two different songs found turning towards the species song despite comparable firing rates on each side. These results have been interpreted as the animal making a comparison of song quality (rather than firing rate *or* latency) on each side. That is, it is argued that the animal can distinguish two sources, recognize how well each resembles the ideal song, and by comparison determine which way to turn. Unless the songs can be separately recognized, it is assumed, the animal could not go directly to one and ignore the other.

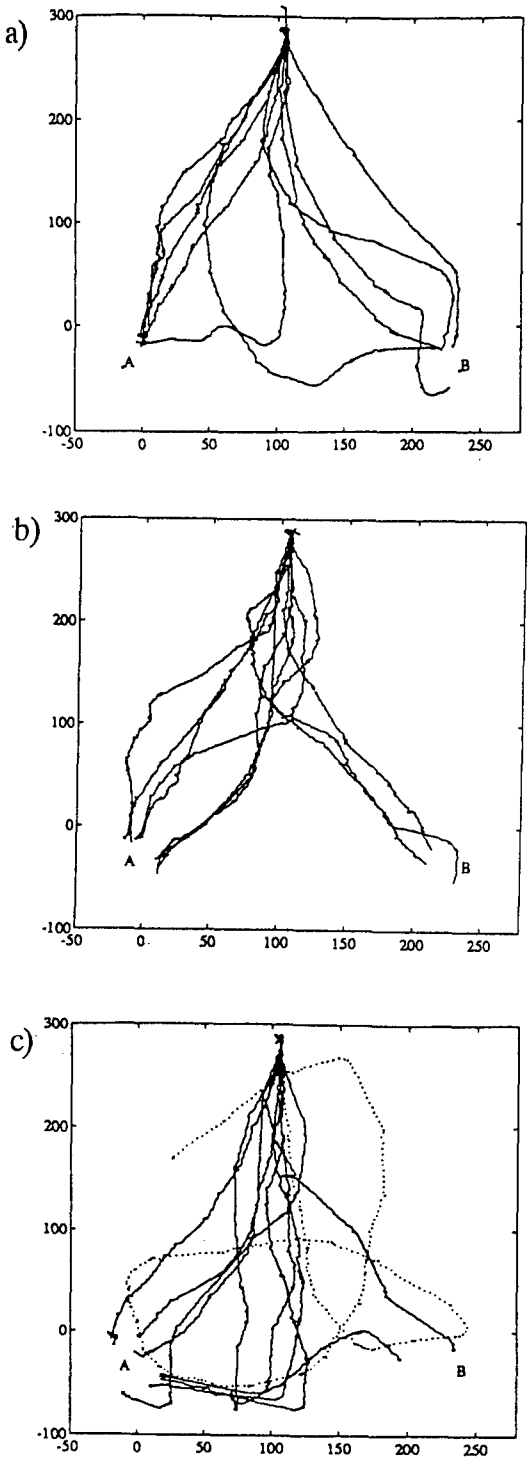
5. Robot choice

We had already determined that in a simple choice situation, the robot algorithm described (figure 3) was sufficient to have it move directly to one out of two sound sources when both were producing ideal songs. The tracks in figure 4 show three conditions: a)with only one speaker producing sound; b)with both speakers producing sound in unison; c)with sound alternating between the speakers. This final condition would not occur naturally; in experiments the cricket shows similar directional errors, moving between the two sound sources and not to one or other.

As the robot was known not to be capable of recognising and deciding between songs, the behaviour in Figure 4b) suggests that recognition-type processing is unnecessary for choice-like behaviour. What follows is a quantitative 'neuroethological' analysis of these results, matching internal and external data records, that enables the following additional questions to be addressed.

- Is it properties of the peripheral sensory system that explain the ability to ignore the second sound source?
- If it is not just the peripheral system, is it the particular mechanism for localization (latency comparison) that explains this behaviour?

Figure 4: robot tracks during phonotaxis from single start point (top of figure) towards speakers (at A and B): a) with either A or B active b) with A and B active c) with A and B alternating



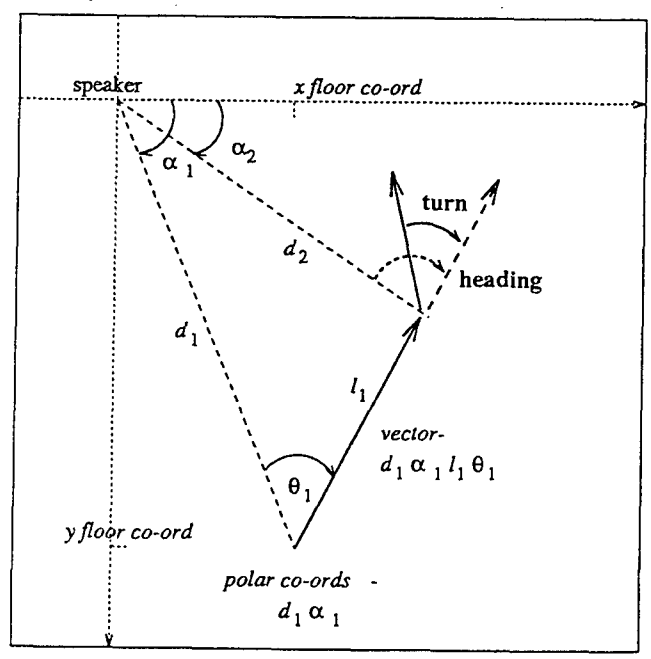
5.1 Analysis Procedure

The paths of the robot from trials illustrated in figure 4 were recorded by an overhead video tracking system. To analyze and compare them it is necessary to do the transformation illlustrated in figure 5. Each path is treated as a set of straight vectors between turns. The vector origin and direction is calculated, in polar co-ordinates, relative to the speaker at which the robot ended its trial. From these vectors the heading of the robot before it turns and the size and direction of the turns can be calculated.

While the robot was performing these trials it internally recorded the values read from the auditory circuit, and whether it was currently performing a turn. Consequently it was possible to align this internal data with the recorded tracks, using the turns as reference points. For each turn, the ten auditory response values that immediately preceded the turn were extracted from this data. It was also possible to recalculate the 'auditory interneuron' (AN) values that would have been generated in the robot by these response values.

In the analysis below, the auditory response and AN values were averaged over the ten readings, which corresponds to roughly 350ms or the length of a burst of song. Thus the left-right difference in average amplitude at the 'ears' and the average activity of the 'interneurons' before a turn could be calculated. Positive values indicate a higher left ear response (or left interneuron activity) than right ear response (or right interneuron activity); negative values the reverse.

Figure 5: Vector transformation of the tracks. The critical values used for further analysis are the ***heading*** *angle relative to the speaker and the* ***turn*** *angle: e.g. in the following section figure 6a) plots turn vs. prior heading for every vector.*



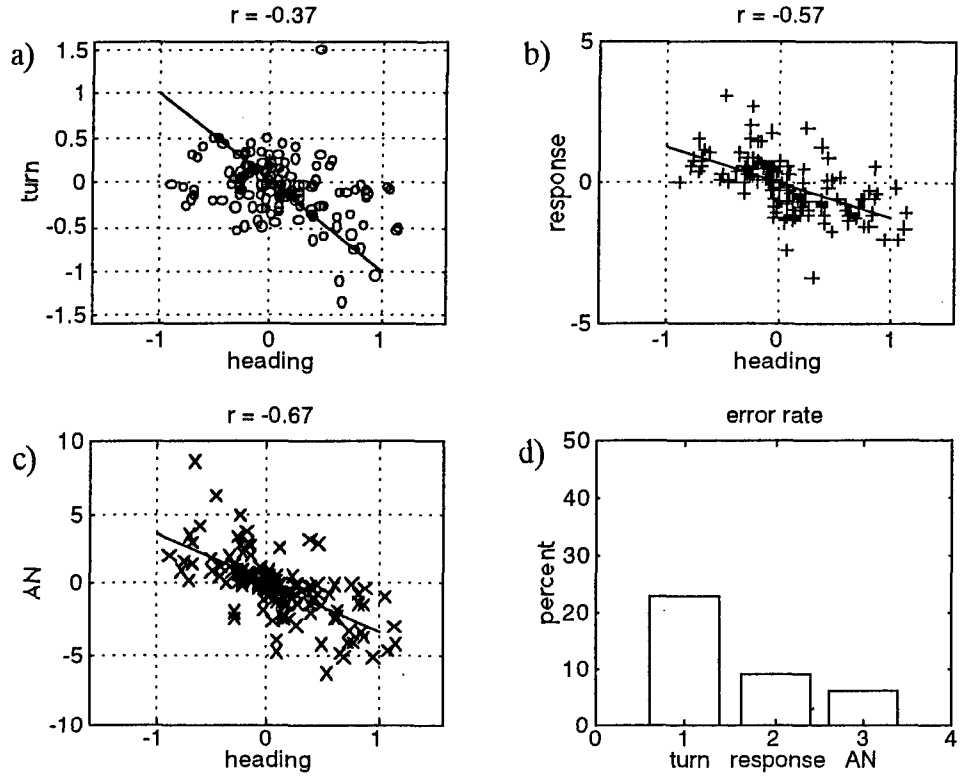


Figure 6: Single sound source: plotting the heading against a) turn size (line is y=-x) b) auditory response (with regression line) c) 'interneuron' (AN) activity (with regression line). d) shows the percentage incorrect e.g. the proportion of turns away from the sound source, or stronger responses on the side furthest from the sound source

5.2 Results

Figure 6: With a single sound source

Figure 6a shows the size and direction of each turn of the robot plotted against the angle it was heading before that turn. In general the turns are corrective, i.e. a turn is roughly the negative of the heading. Perfect behaviour would have all points falling on the line y = -x (drawn on the graph). It is worth noting that the behaviour is, in this sense, far from perfect. 23% of the turns are actually in the wrong direction; and the correlation of turns and heading is -0.37. Nevertheless, judged by the ability to find a speaker by a reasonably direct path, comparable to the paths produced by crickets, the behaviour is perfectly satisfactory. This is an interesting indication of how robust taxis is as a basic mechanism of finding something. More generally, it suggests that a biologically identified neural process that seems rather inaccurate may still potentially be adequate to explain successful behaviour.

Figure 6b shows the auditory response difference plotted against heading angle for each turn, and 6c shows the AN activity difference. The regression line is shown. Both measures are more accurate than the actual turning behaviour, i.e. the auditory response and AN activity are fairly clearly related to the heading, with respective directional error rates of 9% and 6% and correlations of -.57 and -.67. A consequence of this observation is that if the robot *were* to use a firing rate comparison (i.e. the relative activity of left and right interneurons) rather than latency, it would perform more accurate turns. Latency difference comparison (represented by the turn angles which are the immediate output of the latency comparison process) has degraded the information available because it only uses the AN onset and ignores any other fluctuation.

However this does not necessarily suggest that a firing-rate model would be preferable, because the AN difference will be equally accurately related to the heading under any single speaker conditions, including continuous sound or any syllable rate. As was demonstrated in previous work, the latency difference (and turning behaviour dependent on it) disappears under conditions other than the ideal syllable rate; giving rise to apparent recognition of that rate that matches cricket behaviour. A firing rate model would require an additional process to make these distinctions. Again it is interesting to note, for the interpretation of biological data, that the neural signal containing the most information is not necessarily the one driving the behaviour.

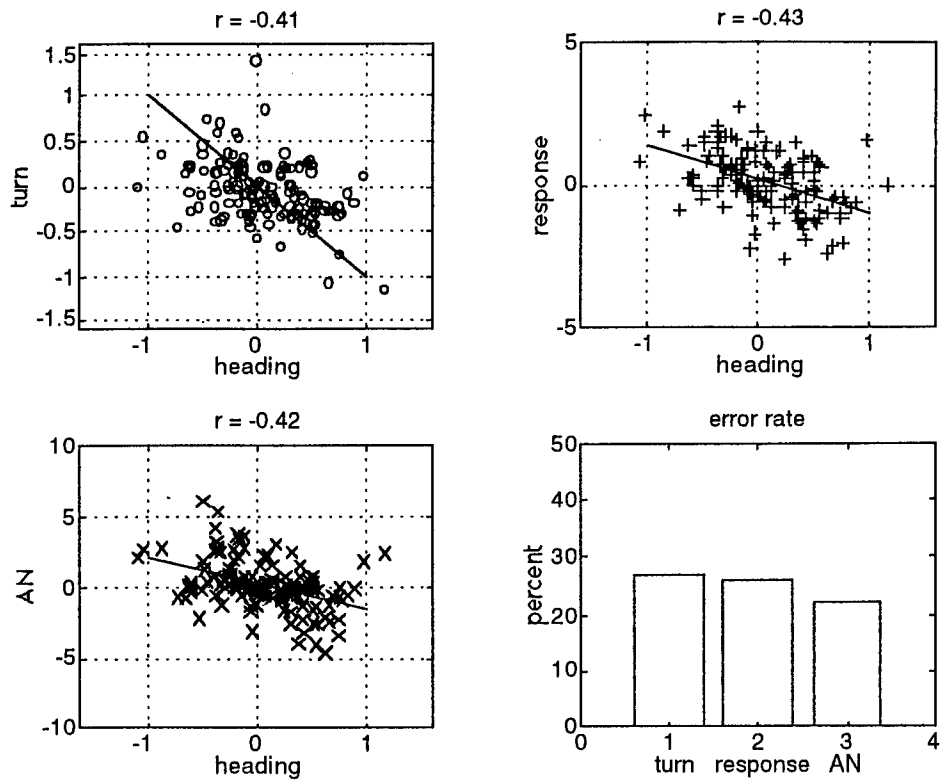


Figure 7: as for figure 6, with two sound sources active in unison

Figure 7: With two sound sources (in unison).

The same relationships plotted in figure 6 are shown for data during trials with two sound sources - the simple choice paradigm - in figure 7. Note that the turn accuracy is comparable to single sound, with a slightly higher correlation of -.41 but also a slightly higher error rate of 27% - neither is statistically different from the previous values[1].

The auditory response accuracy and AN accuracy, on the other hand, have dropped significantly. With sound coming from two places, 26% of the time the response difference is not matched to the heading relative to the speaker being approached, and the AN errors are at 22%. Both percentages are significantly higher than the corresponding single sound measurements; the correlation of -.42 (vs -.67) for AN vs heading is significantly lower; the correlation of -.43 (vs -.57) for response vs heading is not significantly different.

This implies that the 'choice' ability of the robot is not wholly dependent on the peripheral response, as the behaviour has not degraded although the response has. Similarly, the change in accuracy for average activity for AN has not affected the behaviour. A firing rate comparison mechanism would perform less accurately in this situation; a latency comparison mechanism does not.

Figure 8: With two sound sources (alternating)

Obviously the turning behaviour is not wholly *independent* of the auditory response and AN values - rather it appears above that decreases in the average accuracy of the auditory response and AN do not immediately cause decreases in the accuracy of the turns, so the choice behaviour is not directly accounted for by the peripheral or AN sound processing. However if their accuracy decreases sufficiently, the turning response will eventually fail to bring the robot directly toward the sound source; and such a situation is shown in the corresponding graphs for the alternating sound source (figure 8). The turn error rate is 38% and the correlation -.1229. Both response and AN have effectively zero correlation with the heading of the robot (.04).

[1] Percentages were tested for significance difference using a Chi-Square test. Correlations were tested for significant difference by constructing confidence intervals, using Fisher's z transformation (Rohlf and Sokal, 1981). A significance level of 0.05 was used.

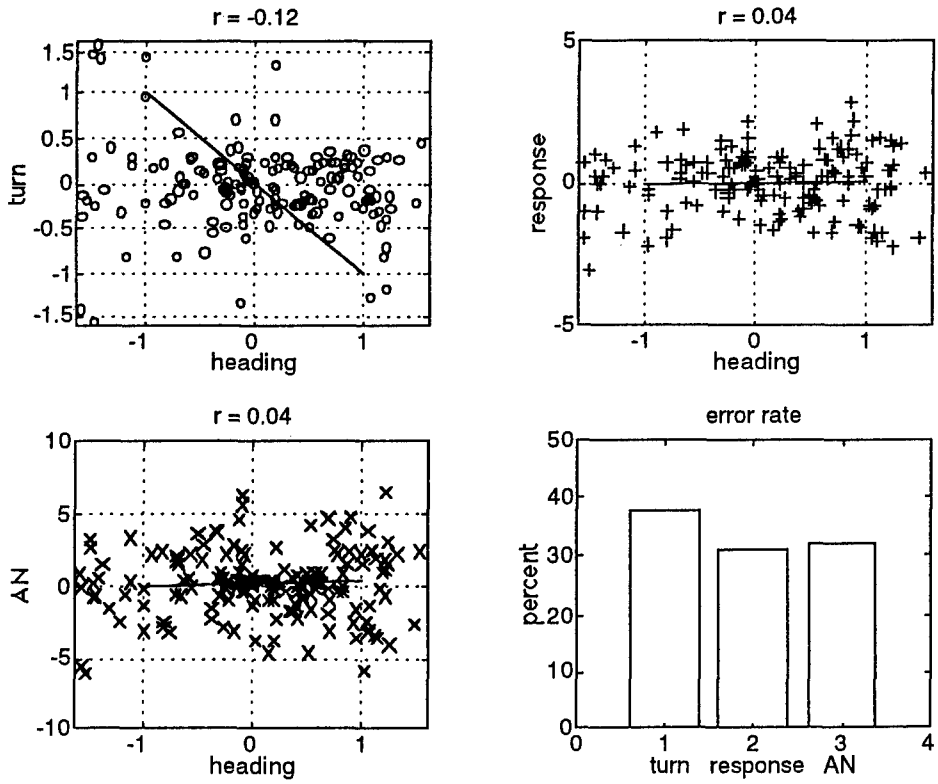


Figure 8: *as for figure 6 with sound alternating between the speakers*

6. Discussion

The results suggest:

- That choice behaviour can occur without a recognition mechanism - the turning behaviour with two sounds in unison is indistinguishable from that with one sound.
- That this ability is not due entirely to the immediate auditory situation of the robot - these differ for one vs. two sounds.
- That the average activity of leaky integrator interneurons does decrease in accuracy in this situation but a latency-based comparison does not.
- That using the average 'firing rate' rather than latency differences would produce more accurate turns with a single sound source, but behaviour would be degraded in the two sound situation.

However it must be emphasised that these results are only for the simplest form of choice behaviour. There are several reasons why the other possibilities cannot be ruled out. First, the existing electronic model of the cricket's auditory processing is slow, and oversimplifies the cancellation function; potentially resulting in a loss of information. Second the neural processes are also highly simplified which may distort the relative effectiveness of latency and firing rate comparisons. Third, the range of choice behaviours displayed by the cricket is much more diverse than the simple case tested so far. Each of these issues is being addressed in current work:

Ears circuit: The new design for the auditory circuit was described in section 3 and figure 2. As well as exploiting the new information about the cricket auditory system to increase the accuracy of the model, this circuit will do the cancellation function in hardware making it fast enough to process real cricket songs (in which syllable repetition is at a rate of 20-30Hz). Thus identical experimental conditions can be used for crickets and robots (see below). However, even this system is still an idealisation of the cricket's auditory system, in that the transfer functions coupling spiracles and tympani measured by Michelson et al. (1994) in the animal show considerable amplitude variation with frequency. A closer emulation of the real system, accomplished by appropriate filter circuitry in the electronic model, may be the subject of further work.

Neural models: The AN-processing in the robot used a simple time averaging filter and thresholding -

$an_t = an_{t-1}*7/8 + response_t$ *,if* $an > 16$, $an = 16$

- to simulate the response in the cricket's auditory interneurons. This does not represent the full complexity of the neural processing. There are two identified pairs of

ascending auditory interneurons, one of which is innervated particularly by the receptors sensitive to calling song frequencies. There is also a pair of 'omega' interneurons that form a mutual inhibition circuit between the two sides and enhance any difference in response between the ascending neurons. The ascending interneurons integrate the receptor response and code the overall response amplitude in spike-rate (0-10) and latency to firing (38-16ms); where spikes are superimposed over increased post-synaptic potential. The recovery time to resting potential is around 15ms. Ambiguous results about the size of syllable gap that these neurons can code were identified while building the robot model and have led to proposals for further neurophysiological recordings (Schildberger, pers. comm.) By capturing some of this complexity of response in the robot model it will be possible to draw further conclusions about the relative contributions of different parts of the firing pattern. There are now many models available for simulating neurons (Arbib, 1995); however constraints of processing power will require a careful choice between such models.

Experiments: To test the robot it will be necessary to do controlled comparisons of different songs. We are proposing to parallel experimentation using the robot and crickets to see whether the same kind of preferences result. The currently available data on cricket choice are simply not sufficient to determine what mechanism underlies it; we believe the robot model will help to lead us to critical experimental designs. It is worth noting already that preference for certain song patterns, rather than purely for higher amplitudes, does not rule out a simple localization model of the kind described here. That is, localization that depends on the pattern is likely to lead to more complex preferences than localization determined only by the 'side with the strongest response'.

7. Conclusions

The results described here and the future work proposed constitute a significant advance in the methodology of robot modelling to test biological hypotheses. Although directed at a specific target system, the tools produced will be more widely applicable. In particular the use of 'neuroethological' analyses of the behaviour in complex real-world situations provides insights into the interpretation of real neuroethological results. It was quite striking to discover the high degree of error in processing that could still result in successful behaviour. It is also worth noting that the signal with least error in one situation is not necessarily the best one to use when a wider behavioural context is considered.

An important motivation for this work is the investigation of minimalism in robot design. We want to discover through experimentation the range of behaviour that a simple control system can produce, if it is coupled to appropriate sensors and actuators in particular environments. This also encourages a close application of Occam's razor in the investigation of biological sensorimotor systems.

References

Arbib, M.A. (ed) (1995) The Handbook of Brain Theory and Neural Networks. MIT Press

Anderson, M.B. (1994) Sexual selection. Princeton University Press

Bailey, W.J., Cunningham, R.J. and Lebel, L. (1990) Song power, spectral distribution and female phonotaxis in the bush cricket *Requena verticalis* (Tettigonide: Orthoptera). Animal Behaviour, 40:33-42

Hedrick (1995) Biotic selection on mate choice for acoustic stimuli in insects and frogs. Proceedings of the 4th International Congress on Neuroethology

Hedrick, A.V. and Dill, L.M. (1993) Mate choice by female crickets is influenced by predation risk. Animal Behaviour 46:193-196

Michelson, A., Popov, A.V. and Lewis, B. (1994) Physics of directional hearing in the cricket *Gryllus bimaculatus*. Journal of Comparative Physiology 175:153-164

Pollack, G. (1986) Discrimination of calling song models by the cricket *Teleogryllus oceanicus*: the influence of sound direction on neural encoding of the stimulus temporal pattern and on phonotactic behaviour. Journal of Comparative Physiology 158:549-561

Popov, A.V. and Shuvalov, V.F. (1977) Phonotacic behaviour of crickets. Journal of Comparative Physiology 119:111-126

Rohlf, F.J. and Sokal, R.R. (1981) Statistical Tables (2nd edition). W.H. Freeman New York

Simmons, L.W. (1988) The calling song of the field cricket *Gryllus bimaculatus (de Geer)* constraints on transmission and its role in intermale competition and female choice. Animal Behaviour 36:380-394

Stabel, J., Wendler, G. and Scharstein, H. (1989) Cricket phonotaxis - localization depends on recognition of the calling song pattern. Journal of Comparative Physiology, 165:165-177

Stout, J.F. and McGhee, R. (1988) Attractiveness of the male *Acheta domestica* calling song to females II The relative importance of syllable period, intensity and chirp rate. Journal of Comparative Physiology, 164:277-287

Webb, B. (1994) Robotic experiments in cricket phonotaxis. From animals to animats 3: Proceedings of the Third International Conference on the Simulation of Adaptive Behaviour MIT Press

Weber, T. and Thorson, J. (1988) Auditory behviour in the cricket IV Interaction of direction of tracking with perceived temporal pattern in split-song paradigms. Journal of Comparative Physiology 163:13-22

Wagner, W.E., Murray, A-M. and Cade, W.H. (1995) Phenotypic variation in the mating preferences of female field crickets *Gryllus integer*. Animal Behaviour 49:1269-1281

Coordination in a six-legged walking system. Simple solutions to complex problems by exploitation of physical properties

Holk Cruse, Christian Bartling, Jeffrey Dean, Thomas Kindermann, Josef Schmitz, Michael Schumm, Hendrik Wagner

Fac. of Biology, University of Bielefeld, Postfach 100131, D-33501 Bielefeld, Germany
e-mail: holk at bio128.uni-bielefeld.de

Abstract

A network for controlling a six-legged, insect-like walking system is proposed. The network contains internal recurrent connections, but important recurrent connections utilize the loop through the environment. This approach leads to a subnet for controlling the three joints of a leg during its swing which is arguably the simplest possible solution. The task for the stance subnet appears more difficult because the movements of a larger and varying number of joints (9-18: three for each leg in stance) have to be controlled such that each leg contributes efficiently to support and propulsion and legs do not work at cross purposes. Already inherently non-linear, four factors further complicate this task: 1) the combination of legs in stance varies continuously, 2) during curve walking, legs must move at different speeds, 3) on compliant substrates, the speed of the individual leg may vary unpredicatably, and 4) the geometry of the system may vary through growth and injury or due to non-rigid suspension of the joints. We show that an extremely decentralized, simple controller, based on a combination of negative and positive feedback at the joint level, copes with all these problems by exploiting the physical properties of the system.

1 Introduction

The study of biological motor systems is difficult because in most such systems, the number of degrees of freedom is normally larger than that necessary to perform the task. This requires the system to select among different alternatives according to some, often context-dependent optimization criteria, which means that the system usually has a high degree of autonomy. Therefore, the experimenter does not have direct control of some important inputs to the motor system. Furthermore, such systems must adapt to complex, often unpredicatable environments. By this "loop through the real world" the unknown properties of the environment affect the behavior of the system. Despite these experimental and theoretical difficulties, the complexity makes the study of motor mechanisms especially challenging, particularly because they illustrate to a high degree the task of integrating influences from the environment, mediated through peripheral, sensory systems, with central processes reflecting the state and needs of the organism. Furthermore, it is exactly the autonomy of animals, expressed in their ability to act adaptively in complex environments without external control and to select among alternative actions in a seemingly efficient way, that has become a focus of interest in recent years in robotics and artificial life as possible models for artificial systems.

In the present article, we shall discuss a motor system for a complex behavior in which hard physical interactions play an essential role. We will describe how the system generates adaptable motor rhythms using a decentralized control system. Furthermore, we will present a work illustrating how the loop through the world can be exploited to simplify computation drastically and to solve otherwise intractable computational problems. The results are based on experimental investigation of the behavior of the biological control system underlying the locomotion of six-legged arthropods in general and the stick insect in particular. Much of the information on the structure of the control system was derived from qualitative and quantitative analysis of the reactions to external disturbances. Such experiments usually provide much more insight into the computational and algorithmic structure of the system than does mere observation of the undisturbed behavior. Although walking, the behavior we discuss here, is sometimes regarded as simple and uninteresting, it involves a very strong and complex interaction with the physical environment. Thus, a study of walking provides a particularly good system for examining mechanisms for integrating autonomous activity with multimodal information from multiple sources including proprioceptors and exteroceptors. A study of the walking system might therefore be of interest not only in itself but also with respect to other, more complicated behaviors where these two types of activity must be integrated.

Behavioral and physiological studies indicate that the flexibility evident in the walking system of the stick insect arises in part because control is distributed among several autonomous centers. The decentralization means that several functional problems which have to be solved can be addressed separately. One problem concerns the way the movement of the individual leg is controlled. The second refers to the coordination between legs. From biological experiments the following general answers can be given. First, each leg has its own control system which generates rhythmic step movements (v. Holst 1943, review Graham 1985, Wendler 1964). The behavior of this control system corresponds to that of a relaxation oscillator in which the change of state, the transition between stance and swing, is determined by thresholds based on leg position. Second, several coordinating mechanisms couple the movement of the legs to produce a proper gait (Cruse 1990, Dean 1991a,b, 1992a,b). These results are briefly summarized in the next section (for details see Cruse et al. 1995a,b).

2 Control of the step rhythm of the individual leg

The step cycle of the walking leg can be divided into two functional states. During stance, the leg is on the ground, supports the body and, in the forward walking animal, moves backwards with respect to the body. During swing, the leg is lifted off the ground and moved in the direction of walking to where it can begin a new stance. The anterior transition point, i.e., the transition from swing to stance in the forward walking animal, has been called the anterior extreme position (AEP) and the posterior transition point has been called the posterior extreme position (PEP). Differences in the constraints acting during the two states and in the conditions for their termination suggest that the leg controller may consist of separate control networks (although this separation may only be justified on a logical level and not reflect a morphological separation in the nervous system). In the terminology of Minsky (1985), these networks may be regarded as "agents". Using this term, this relaxation oscillator making up the step pattern generator is assumed to consist of two agents--a swing network controlling the swing movement, and a stance network controlling the movement of the leg during stance (Fig. 1b). The transition between swing and stance is controlled by a third agent, the selector network (Fig. 1b). The swing network and the stance network are always active, but the selector network determines which of the two agents controls the motor output. To match experimental results which failed to demonstrate a robust central pattern generator producing strong intrinsic rhythms (Bässler and Wegner 1983), this selection is done on the basis of sensory input. The selector net consists of a two-layer feedforward net with positive feedback connections in the second layer. These positive-feedback connections serve to stabilize the ongoing activity, namely stance or swing.

The three most important coordinating mechanisms influence the selector net so as to modulate the beginning of a swing movement, and therefore the end-point of a stance movement (PEP). These influences will not be described here in detail (see Cruse et al. 1995b, Fig. 3). The end of the swing movement in the animal is modulated by a single, caudally directed influence (Cruse et al. 1995b, Fig. 3, no. 4) depending on the position of the next rostral leg. This mechanism is responsible for the targeting behavior--the placement of the tarsus at the end of a swing close to the tarsus of the adjacent rostral leg.

In this paper we first explain the architecture of the present swing net, which is still simpler than earlier versions (Cruse et al. 1995a,b). Second, we present an extremely simple solution for the stance net which we consider to be a fundamentally new approach.

3 Control of the swing movement

The task of finding a network that produces a swing movement seems to be easier than finding a network to control the stance movement because a leg in swing is mechanically uncoupled from the environment and therefore, due to its small mass, essentially uncoupled from the movement of the other legs. The geometry of the leg is shown in Fig. 1a. The coxa-trochanter and femur-tibia joints, the two distal joints, are simple hinge joints with one degree of freedom corresponding to elevation and to extension of the tarsus, respectively. The subcoxal joint is more complex, but most of its movement is in a rostrocaudal direction around the nearly vertical axis. The additional degree of freedom allowing changes in the alignment of this axis is little used in normal walking (Cruse and Bartling 1995), so the leg can be considered as a manipulator with three degrees of freedom for movement in three dimensions. Thus, the control network must have at least three output channels, one for each leg joint. As has been shown by Cruse et al. (1995a,b), a simple, two-layer feedforward net with three output units and six input units can produce movements which closely resemble the swing movements observed in walking stick insects (Cruse and Bartling 1995). In the simulation, the three outputs of this net, interpreted as the angular velocities $d\alpha/dt$, $d\beta/dt$, and $d\gamma/dt$, are fed into an integrator (not shown in Fig. 1b), which corresponds to the leg itself in the animal, to obtain the joint angles. The actual angles are measured and fed back into the net. Besides the input to the swing net provided by three sensors measuring the actual angles of the three leg joints, three other input units (α_t, β_t, γ_t) represent the target of the swing movement, i.e., the leg position that should be achieved at the end of the return stroke (targeting influence). These values might correspond either to a fixed AEP, or to a variable AEP depending on information coming from the next anterior leg (see "target net", below).

Using such a linear system, the form of the leg trajectory closely approximated the biological data, but the velocity profile did not. The velocity was highest at the start and then decreased monotonically. Therefore, a nonlinear compensation (Fig. 1b, NL) was introduced; the output of all three motor units was multiplied by a common factor depending on the actual leg position relative to the target position. Recent work further simplified the weight matrix of this network compared to the earlier versions. These matrices are shown in Table 1 in the same format as for the swing net in Fig. 1b.

This system with only 8 or 9 nonzero weights, which most probably represents the simplest possible network, is able to generalize over a considerable range of untrained situations, a further advantage of the network approach. Furthermore, the swing net is remarkably tolerant with respect to external disturbances. The learned trajectories represent a kind of attractor to which the disturbed trajectory returns. This compensation for disturbances occurs because the system does not compute explicit trajectories, but simply exploits the physical properties of the world. Errors resulting from such disturbances, of course, will be greater, the larger the disturbance and the later it occurs within the return stroke. However, such errors in leg placement do not strongly influence the overall behavior of the walking system as long as the leg successfully contacts the ground.

This ability to compensate for external disturbances permits a simple extension of the swing net in order to simulate an avoidance behavior observed in insects. When a leg strikes an obstacle during its swing, it initially attempts to avoid it by making a short retraction and elevation and then renewing its swing forward from this new position. In the augmented swing net, an additional input similar to a tactile or force sensor signals such mechanical disturbances (Fig. 1b, m.d.). This unit is connected by fixed weights to the three motor units in such a way as to produce a brief retraction and elevation seen in the avoidance reflex. These weights are shown in the top row of Table 1.

In the model, the targeting influence reaches the leg controller as the input to the swing net (Fig. 1b). It was shown earlier by Dean (1990) that a simple feedforward net with three hidden units and logistic activation functions (Fig. 1b, "target net") can transform the values of the joint angles of a rostral leg to joint angles for the next caudal leg such that the tarsi of the two legs are at the same position. This is possible with an exactness corresponding to that found in biological experiments (Cruse 1979, Dean and Wendler 1982), showing that an approximation can provide adequate control and replace an exact solution of coordinate transformations involved in the direct and inverse kinematic problem. Physiological recordings from local and intersegmental interneurons (Brunn and Dean 1994) support the hypothesis that a similar approximate algorithm is implemented in the nervous system of the stick insect.

Because a network solution for controlling leg movement during stance seemed at first sight to be quite complicated, the initial simulations use the inverse kinematics to find joint configurations to move the tarsus along a straight line parallel to the long axis of the body (alternative, network solutions to the control of stance movement are discussed below). This model shows a proper coordination of the legs when these are walking at different speeds on a horizontal plane (Cruse et al. 1995a,b). Steps of ipsilateral legs are organized in triplets forming "metachronal waves", which proceed from back to front, whereas steps of the contralateral legs on each segment step approximately in alternation. With increasing walking speed, the typical change in coordination from the tetrapod to a tripod-like gait is found. For slow and medium velocities the walking pattern corresponds to the tetrapod gait with four or more legs on the ground at any time and diagonal pairs of legs stepping approximately together; for higher velocities the gait approaches the tripod pattern with front and rear legs on each side stepping together with the contralateral middle leg. The coordination pattern is very stable. For example, when the movement of one leg is interrupted briefly during the power stroke, the normal coordination is regained immediately at the end of the perturbation. Note that the temporal sequence of the activities of the legs is implicitely determined by the connections between the leg controllers, in a manner similar to the approach described by Maes (1991).

A particularly difficult problem for the walking system is to begin a walk from different starting configurations, i.e., positions of the six legs relative to the body. In earlier versions of the network model (Cruse et al. 1993a), many starting configurations led to unstable positions during the first steps. This means that the center of gravity was not supported by the polygon formed by the legs standing on the ground. For the insect, this may not be a serious problem because the supporting legs can attach to most natural substrates and pull as well as push; for a machine, ground contact is usually passive, so it is important to maintain an adequate arrangement of supporting legs. To improve the situation, the weights describing the strength of the different coordinating mechanisms were varied using the technique of genetic algorithms. With the best set of weights found to date, 92% of the simulated sequences contain no instabilities and less than 1% include instabilities amounting to more than 0.4 of the swing duration. These values hold for moderate walking speeds (ratio of swing/stance duration: 0.4:1) starting from randomly selected leg configurations. At lower walking speeds (ratio of swing-to-stance duration: 0.2/1), no walks contain instabilities. At higher walking speeds the percentage of stable walks decreases.

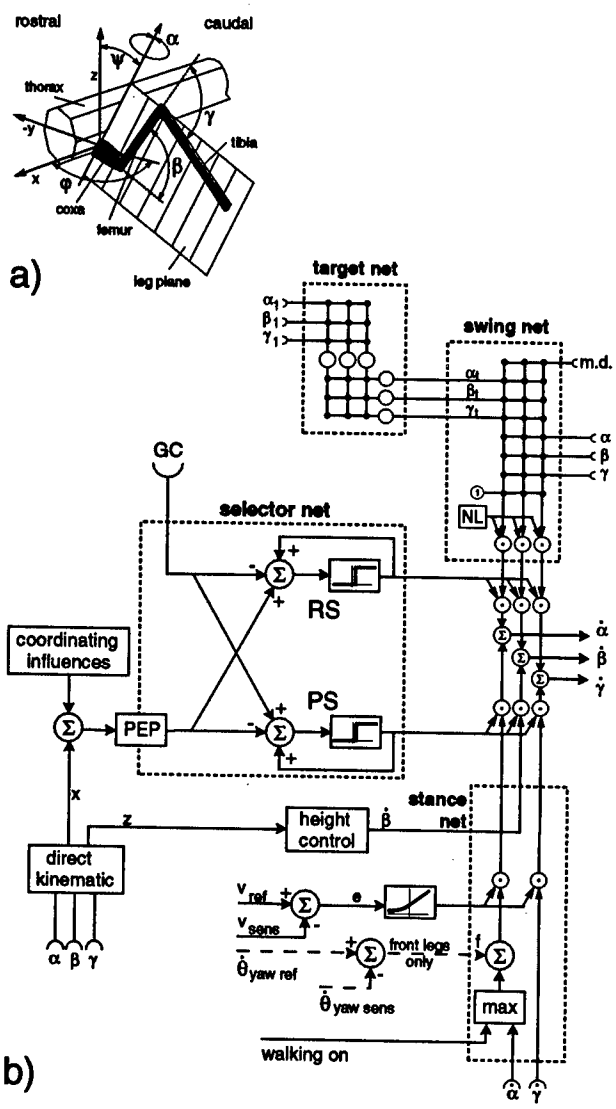


Fig. 1. Summary of leg geometry and the control networks for the model. (a) Schematic drawing of a mechanical model of a stick insect leg showing the arrangement of the joints and their axes of rotation. (b) The leg controller consists of three parts: the swing net, controlling the leg movement during swing, the stance net for the control of the stance movement, and the selector net which determines whether the swing or the stance net can control the motor output, i.e., the velocity of the three joints α, β, and γ. The selector net contains four units: the PEP unit signalling posterior extreme position, the GC unit signalling ground contact, the RS unit controlling the return stroke (swing movement), and the PS unit controlling the power stroke (stance movement). The target net transforms information on the configuration of the anterior target leg, given by the angles α_1, β_1, γ_1, into angular values for the next caudal leg (α_t, β_t, γ_t) which place the two tarsi close together. These desired final values and the current values of the leg angles, α, β, and γ, are input to the swing net together with bias inputs and a sensory input (m.d.) which is activated by an obstruction blocking the swing and thereby initiates an avoidance movement. NL is the nonlinear compensation. The stance net uses the β joint for height control via a negative feedback system. Walking velocity is controlled by a negative feedback system which compares v_{ref} and v_{sens} , passes the error signal, e, through a nonlinear characteristic (see text) and influences the α and the γ gain. The α channels of the front legs are further subject to a yaw control (angle θ) which stabilizes straight and curved walking. The sign of the factor f is opposite for the left and right front legs. Walking is switched on by another central influence which passes through a maximum detector and influences the α channels of all legs

Table 1 shows the weights (x 100) used for the swing nets of front, middle and rear legs. The weight matrices are organized in the same manner as the nodes in the swing net shown in Fig. 1b (i.e., the top row refers to the m.d. input, the bottom row to the bias).

front leg			middle leg			hind leg		
45	-25	-50	65	15	-100	35	40	-100
54	0	0	29	32	0	32	55	0
0	8	0	0	15	0	0	33	0
0	0	20	0	0	25	0	0	34
-53	-28	0	-22	-50	0	-26	-55	0
0	-45	0	0	-20	0	0	20	0
0	-42	-25	0	0	-25	0	9	-33
0	0	0	0	0	0	0	0	0

4 Control of the stance movement and coordination of supporting legs

The control solution proposed above for the stance movement is feasible for straight walking on a flat surface. In more natural situations, the task of controlling the stance movements of all the legs on the ground involves several problems. It is not enough simply to specify a movement for each leg on its own: the mechanical coupling through the substrate means that efficient locomotion requires coordinated movement of all the joints of all the legs in contact with the substrate, that is, a total of 18 joints when all legs of an insect are on the ground. However, the number and combination of mechanically coupled joints varies from one moment to the next, depending on which legs are lifted. The task is quite

nonlinear, particularly when the rotational axes of the joints are not orthogonal, as is often the case in insects, particularly for the basal leg joint. A further complication occurs when the animal negotiates a curve because then the different legs move at different speeds.

In machines, these problems can be solved using traditional, though computationally costly, methods, because these approaches consider the ground reaction forces of all legs in stance and seek to optimize some additional criteria, such as minimizing the tension or compression exerted by the legs on the substrate (Weidemann et al. 1993). Due to the nature of the mechanical interactions and inherent in the search for a globally optimal control strategy, such algorithms require a single, central controller; they do not lend themselves to distributed processing. This makes real-time control difficult, even in the still simple case of walking on a rigid substrate.

Further complexities arise in more complex, natural walking situations, making solution difficult even with high computational power. These occur, for example, when an animal or a machine walks on a slippery surface or on a compliant substrate, such as the leaves and twigs encountered by stick insects. Any flexibility in the suspension of the joints further increases the degrees of freedom that must be considered and the complexity of the computation. Further problems for an exact, analytical solution occur when the length of leg segments changes during growth or their shape changes through injury. In such cases, knowledge of the geometrical situation is incomplete, making an explicit calculation difficult, if not impossible.

Despite the evident complexity of these tasks, they are mastered even by insects with their "simple" nervous systems. Therefore, there has to be a solution that is fast enough that on-line computation is possible even for the slow neuronal systems. How can this be done? Several authors (e.g., Gibson 1966 for perception, Brooks 1991 for controlling action) have pointed out that some relevant parameters do not need to be explicitly calculated by the nervous system because they are already available in the interaction with the environment. This means that, instead of an abstract calculation, the system can directly exploit the dynamics of the interaction and thereby avoid a slow, computationally exact algorithm. To solve the particular problem at hand, we propose to replace a central controller with distributed control in the form of local positive feedback (Cruse et al. 1995c, Schmitz et al. 1995). Compared to earlier versions (Cruse et al. 1993a, 1995a,b), this change permits the stance net to be radically simplified. The positive feedback occurs at the level of single joints: the position signal of each is fed back to control the motor output of the same joint (Fig. 1b, stance net). How does this system work? Let us assume that any one of the joints is moved actively. Then, because of the mechanical connections, all other joints begin to move passively, but in exactly the proper way. Thus, the movement direction and speed of each joint does not have to be computed because this information is already provided by the physics. The positive feedback then transforms this passive movement into an active movement.

There are, however, several problems to be solved. The first is that positive feedback using the raw position signal would lead to unpredictable changes in movement speed, not the nearly constant walking speed which is usually desired. This problem can be solved by introducing a kind of band-pass filter into the feedback loop. The effect is to make the feedback proportional to the angular velocity of joint movement, not the angular position. In the simulation, this is done by feeding back a signal proportional to the angular change over the preceding time interval, i.e., the current angle minus the angle at the previous time step. The second problem is that using positive feedback for all three leg joints leads to unpredictable changes in body height, even in a computer simulation neglecting gravity. A physical system, of course, would be pulled downward by gravity and the positive feedback would accelerate this movement. In summary, a system with positive feedback at all joints does not maintain a constant body height even in the absense of gravity; it is even less able to do so under the influence of gravity. Such control, of course, is an essential characteristic of an efficient walking system.

During walking, body height of the stick insect is controlled by a distributed system in which each leg acts like an independent, proportional controller (Cruse 1976, Cruse et al. 1993b). Thus, no master height controller is necessary; the only central information is the invariant reference value for each leg. For an individual leg, however, maintaining a given height via negative feedback appears at odds with the proposed local positive feedback for forward movement. How can both functions be fulfilled at the same time? To solve this problem we assume that during walking positive feedback is provided for the α joints and the γ joints, (Fig. 1, stance net), but not for the β joints. The β joint is the major determinant of the separation between leg insertion and substrate, which determines body height. The action of the γ joint in extending the leg is less important. Thus, in the control scheme for walking proposed here, the β joint remains under classical negative feedback as in the standing animal. In this way, it is possible to solve all the problems mentioned above in an easy and computationally simple way.

It was previously shown that the system can perform one complete stance movement when all six legs stand on the ground (Cruse et al. 1995c). In this previous work, the mechanics of the six legged system were simulated using a specially designed recurrent network (MMC-net, Steinkühler et al. 1995). However, it was not clear whether this system can produce a straight walk over many steps and whether it can negotiate curves.

To investigate these questions, we combined the MMC net for the six-legged structure with the neural network controller described above for the walking machine. Preliminary tests showed that continuous walks are possible but several problems occur. First, the system did not walk straight but moved in a slight curve as a result of a tendency to yaw (rotate around the vertical body axis). To prevent this yawing, a supervisory system was introduced which, in a simple way, simulates optomotor mechanisms for course stabilisation that are well-known from insects and have also been applied in robotics (Franceschini et al. 1992). This supervisory system uses information on the rate of yaw ($\dot{\theta}_{\text{yaw sens}}$, Fig. 1b, stance net), such as could be provided by visual movement detectors. It is based on negative feedback of the deviation between the desired turning rate and the actual change in heading over the last time step. The error signal controls additional impulses to the α joints of the front legs which have magnitudes proportional to the deviation and opposite signs for the right and left sides. With this addition and $\dot{\theta}_{\text{yaw ref}}$ set to zero, the system moves straight (Fig. 2a) with small, side-to-side oscillations in heading such as can be observed in walking insects. To simulate curve walking, the reference value is given a small positive or negative bias to determine curvature direction and magnitude. Fig. 2b shows an example of a left turn.

A second problem inherent in using positive feedback is the following. Let us assume that a stationary insect is pulled backward by gravity or by a brief tug from an experimentor. With positive feedback control as described, the insect should then continue to walk backwards even after the initial pull ends. This has never been observed. Therefore, we assume that a supervisory system exists which is not only responsible for switching on and off the entire walking system (Schmitz and Haßfeld 1989), but also specifies walking direction (normally forward). This influence is represented by applying a small, positive input value (Fig. 1b, walking on) which replaces the sensory signal if it is larger than the latter (the box "max" in Fig. 1b, stance net).

Finally, we have to address the question of how walking speed is determined in such a positive feedback controller. Again, we assume a central value which represents the desired walking speed v_{ref}. This is compared with the actual speed, which could be measured by visual inputs or by monitoring leg movement. This error signal is subject to a nonlinear transformation {$y = (1+e)$ for $e \geq 0$ and $y = 1/(1-e)$ for $e < 0$} and then multiplied with the channels providing the positive feedback for all α and γ joints of all six legs (Fig. 1b, stance net). No examples for different walking speeds are given.

Sometimes, the following unexpected behavior was observed. In the course of a long, apparently stable walk, the system stumbled for as yet undetermined reasons and fell to the ground, but in all cases, the system stood up by itself and resumed proper walking. This happened even when the fall placed the six legs in an extremely disordered arrangement. This means that the simple solution proposed here also eliminates the need for a special supervisory system to rearrange leg positions after such an emergency.

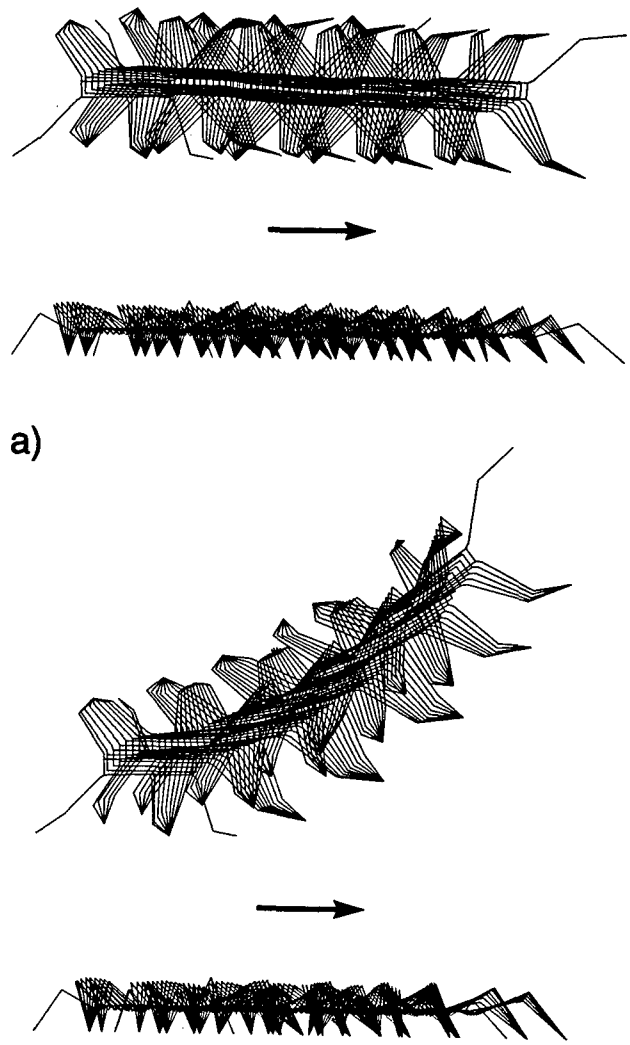


Fig. 2. Simulated walk by the basic six-legged system with negative feedback applied to all six β joints and positive feedback to all α and γ joints as shown in Fig. 1b. Movement direction is from left to right (arrow). Leg positions are illustrated only during stance and only for every second time interval in the simulation. Each leg makes about five steps. Upper part: top view, lower part: side view.

(a) Straight walking ($\dot{\theta}_{\text{yaw ref}} = 0$).

(b) curved walking ($\dot{\theta}_{\text{yaw ref}} \neq 0$).

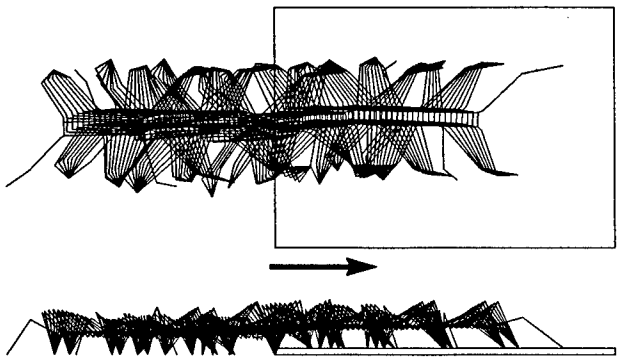


Fig. 3. Straight walk over a step. For further explanations see Fig. 2 and text

The walks shown in Figs. 2a,b are computed for the system walking on a horizontal flat surface. Intial tests indicate that the system with no additional modification can often overcome such steps successfully, but additional tests are in progress. To illustrate the behavior on an uneven surface, Fig. 3 shows the system walking over a step. The step height corresponds to one third of the average body-ground distance.

5 Discussion

In order to produce an active behavior, i.e., a time-varying motor output, a system needs recurrent connections at some level. A pacemaker neuron incorporates these recurrent influences within a single neuron, one which can produce simple rhythms on its own. More interesting are networks -- with or without pacemakers -- where the recurrent connections involve non-linear interactions among two or more units. In such systems, recurrent connections may occur in two ways. They may occur as internal connections, as in the various types of artificial recurrent networks (e.g. Hopfield nets or Elman nets). Alternatively, if a simple system with a feedforward structure is situated in a real or simulated environment, information flow via the environment serves as an "external" recurrent connection because any action by the system changes its sensory input, thereby closing the loop.

We have described a system containing recurrent connections of both types which can be used to control hexapod walking. The system is constructed of elements representing simplified artificial neurons with nonlinear activation functions. There is no central controller but each leg has its own, independent control system able to generate quasi-rhythmic stepping movements. These six systems are coupled by a small number of connections forming a recurrent network. Together with the ever present external recurrent connections - during walking the legs are mechanically coupled via the substrate - this recurrent system produces a proper spatio-temporal pattern corresponding to different gait patterns found in insects. One subsystem, the swing net, consists of an extremely simple feedforward net which exploits the recurrent information via the sensorimotor system as the leg's behavior is influenced by the environment to generate swing movements in time. The system does not compute explicit trajectories; instead, it computes changes in time based on the actual and the desired final joint angles. Thus, the control system exploits the physical properties of the leg. This organization permits an extremely simple network and responds in an adaptive way to external disturbances.

Another result is an increase in the number of starting configurations for which walking begins with no instability. Interestingly, problems with instability at the start of walking have not been reported by other investigators of six-legged walking systems (e.g., Amendt and Frik 1994, Beer et al. 1993, Berns et al. 1994, Brooks 1989, Espenschied et al. 1993, 1994, Pfeiffer et al. 1994). Four reasons might explain this fact. First, the initiation of walking may not have been of interest to these authors, so only safe starting positions were tested. Second, in other machines, a special program may have rearranged the legs into safe positions before initiating walking. Third, relatively small step amplitudes have been used. As our simulations have shown, decreasing step amplitude decreases the number of instable starting positions. In the stick insect the step amplitude of a single leg corresponds to about 2/3 of the distance between the base of the front leg and that of the hind leg. Where step amplitude is mentioned in the papers cited, they are considerably smaller. Fourth, only slow walking speed may have been tested. As mentioned above, the number of instabilities decreases with decreasing walking speed.

A more severe problem has to be solved by the stance system, the second subsystem of the leg controller. During stance, the movements of many joints (9-18) have to be controlled in order that each leg contributes efficiently to support and propulsion and that the legs do not work at cross purposes during walking. This task is already inherently non-linear, but several factors make it still more difficult. First, the number and combination of legs on the ground varies during stepping. Second, each leg has to move at a different speed during curved walking. Third, important effects are unpredictable: for example, when walking on soft ground, the speed of the individual leg changes in an unforseeable way. Finally, even the geometry of the system may change due to non-rigid suspension of the joints, or due to growth or injury. Whereas the predictable changes are solvable, although they make a centralized controller to mediate the cooperation seem unavoidable at first sight, the unpredictable effects make a computational solution still more difficult if not impossible.

The first two points are solved by different approaches in different ways. The simplest solution is provided by Brooks (1989). Here the two-joint legs are simply driven by high torques which means that considerable forces/stresses occur across the body. The situation is similar for a robot presented by Beer and colleagues (Espenschied et al. 1993) and for several simulations (e.g. Beer et al. 1992, Cruse et al. 1995a,b). In principle the same effects occur when, as done by Pfeiffer et al. (1994), a predetermined trajectory for each joint is calculated and followed using a PID-controller for the motor. Another solution is to use P-controllers so that each joint has some spring-like properties (Espenschied et al. 1994, 1996). In the latter case the situation is further improved by incorporating a mechanical spring into the distal segment to provide passive accomodation in length. Similarly, in a more recent robot of Brooks and colleagues (Ferrell 1993), angle trajectories are calculated and a servocontroller is used to determine the motor torque. In each of these cases some kind of central controller, at least at the level of the individual legs, is necessary to determine the joint movements corresponding to the desired tarsus trajectory. None of the prior studies address the latter two points. Solving these much more complex problems in an analytical way requires a global controller with precise information about the current geometry of the system as it might be affected by injury, flexible joint suspension, ground compliance and slippage, and so on. The distributed system presented here, in contrast, can handle these problems with no increase in computational complexity.

It appears that solving these complex tasks requires quite a high level of "motor intelligence". However, this does not require a complicated, or a centralized control system. By contrast, we have shown in simulations that an extremely decentralized solution copes with all these problems and, at the same time, allows a very simple structure for the local controllers. No neuronal connections among the joint controllers, even among those of the same leg, are necessary during stance. The necessary constraints are simply provided by the physical connections of the joints. Thus, the system controlling the joint movements of the legs during stance is not only "intelligent" in terms of its behavioral properties, but also in terms of simplicity of construction: the most difficult problems are solved by the most decentralized control structure. This simplification is possible because the physical properties of the system and its interaction with the world are exploited to replace an abstract, explicit computation. Thus, "the world" is used as "its own best model" (Brooks 1991). Due to its extremely decentralized organization and the simple type of calculation required, this method allows for a very fast computation which can be further accelerated by implementation on a parallel computer.

Some central commands from a superior level are still required. These are necessary to determine the beginning and end of walking as well as its speed and direction. However, these commands do not have to be precisely adjusted to the particular configurations and states of all the legs; an approximate specification is sufficient for the positive feedback control. For example, turning is possible without having to send a corresponding command to every leg; appropriate commands to both front legs are sufficient. The solution proposed for curved walking is definitely simpler than it is in biological systems. In stick insects (Dean, in prep.) and in crayfish (Cruse and Silva Saavedra 1996) the positions of AEP and/or PEP are changed during curved walking. This is not yet implemented in the simulation but will probably improve its turning behavior. Once these additions have been made we expect to implement the control architecture on the TUM walking machine (Pfeiffer et al. 1994).

It is not clear how this supervisory system works within the biological system, but there are some hints which will guide further experiments. The idea of using positive feedback to support active leg movement goes back to experimental findings of Bässler (1976). He showed that, in an active animal, elongation of the femoral chordotonal organ of the femur-tibia joint, which occurs during joint flexion, leads to an inhibition of activity in the extensor muscle, facilitating rather than opposing continued flexion. In a series of further investigations Bässler (for review see Bässler 1993) have made it very probable that this inhibition of the extensor muscle and the concommittant activation of the antagonistic flexor muscle result from positive feedback. However, as negative feedback is seen in a number of other experimental situations, the interpretation is difficult (for discussion see Cruse et al. 1995b). One resolution of this contradiction would be to interpret the results as "phenomenological" positive feedback generated in a negative feedback system. By this we mean that the movement induced a shift in the reference value for the negative feedback system causing it to assist rather than oppose the change in position. This interpretation was ruled out, though, by an experiment of Schmitz (Schmitz et al. 1995) showing directly the existence of logical positive feedback.

Two applications of positive feedback have been discussed. The use of positive feedback in the context of motor control has been considered previously with respect to recurrent circuits within the brain (Houk et al. 1993). The interpretation advanced is that the positive feedback within motor and premotor centers serves to sustain ongoing activity. Similar results have been described for the swimming system of the leech (Kristan et al. 1992). This basically corresponds to the simple circuit used in the selector net (Fig.1b) to sustain power stroke or return stroke activity in the walking leg (Qualitatively, the effect is the same: ongoing patterns of movement are sustained through the recurrent connections. Sensory pathways are not part of the positive feedback

although sensory inputs may effect changes from one pattern to another. Furthermore, the positive feedback influences the dynamics of transitions among patterns and the selection of one pattern over another, but it does not influence the parameters of the pattern. An uncontrolled increase of the output signal is prevented by the Heaviside activation function which permits only a 0 or 1 output. In the stance net presented here, in contrast, the positive feedback loop includes elements outside the nervous system, i.e., it includes the sensorimotor loop through the world, and it modulates the pattern itself by affecting relative levels of activity in different joints. Therefore, graded output values are possible and desired. The band-pass filter in the positive feedback pathway prevents uncontrolled increases in activation.

Acknowledgements

This work was supported by the Körber Foundation, by BMFT (grant no. 01 IN 104 B/1), and DFG (grant no. Cr 58/8-2).

References

Amendt, O., Frik, M. (1994): Motion pattern generation for a six-legged walking machine using neural networks. 6th German-Japanese Seminar on Nonlinear Problems in Dynamical Systems - Theory and Application, Chino, Nagano, Japan

Bässler, U. (1976): Reversal of a reflex to a single motor neurone in the stick insect Carausius morosus. Biol. Cybern. 24, 47-49

Bässler, U. (1993): The femur-tibia control system of stick insects - a model system for the study of the neural basis of joint control. Brain Res. Rev. 18, 207-226

Bässler, U., Wegner, U. (1983): Motor output of the denervated thoracic ventral nerve cord in the stick insect Carausius morosus. J. exp. Biol. 105, 127-145

Beer, R.D., Chiel, H.J., Quinn, R.D., Espenschied, K.S., Larsson, P. (1992): A distributed neural network architecture for hexapod robot locomotion. Neural Computation 4, 356-365

Berns, K., Piekenbrock, St., Dillmann, R. (1994): Learning control of a six-legged walking machine. Proceedings of the 5th Intern. Symp. on Robotics and Manufacturing. M. Jamashidi, Ch. Ngyuen, R. Lumia, J, Yuh (eds.), Vol. 5, pp. 29-34. ASME Press, New York

Brooks, R.A. (1989): A robot that walks: emergent behavior from a carefully evolved network. Neural Computation 1, 253-262

Brooks, R.A. (1991): Intelligence without reason. IJCAI-91, Sydney, Australia, 569-595

Brunn, D., Dean, J. (1994): Intersegmental and local interneurones in the metathorax of the stick insect, Carausius morosus. J. Neurophysiol. 72, 1208-1219

Cruse, H. (1976): The control of the body position in the stick insect (Carausius morosus), when walking over uneven surfaces. Biol. Cybern. 24, 25-33

Cruse, H. (1979): The control of the anterior extreme position of the hindleg of a walking insect. Physiol. Entomol. 4, 121-124

Cruse, H. (1990): What mechanisms coordinate leg movement in walking arthropods? Trends in Neurosciences 13, 15-21

Cruse H., Bartling, Ch. (1995): Movement of joint angles in the legs of a walking insect, Carausius morosus. J. Insect Physiol. 41, 761-771

Cruse, H., Bartling, Ch., Cymbalyuk, G., Dean, J., Dreifert, M. (1995a): A modular artificial neural net for controlling a six-legged walking system. Biol. Cybern. 72, 421-430

Cruse, H., Bartling, Ch., Brunn, D.E., Dean, J., Dreifert, M., Kindermann, T., Schmitz, J. (1995b): Walking: a complex behavior controlled by simple systems. Adaptive Behavior 3, 385-418

Cruse, H., Bartling, Ch., Kindermann, T. (1995c): High-pass filtered positive feedback for decentralized control of cooperation. In: F. Moran, A. Moreno, J.J. Merelo, P. Chacon (eds.), Advances in Artificial Life, pp. 668-678. Springer 1995

Cruse, H., Müller-Wilm, U., Dean, J. (1993a): Artificial neural nets for a 6-legged walking system. In: From animals to animats 2. J.A. Meyer, H.L. Roitblat, S.W. Wilson (eds.). pp. 52-60. MIT Press, Cambridge, MA

Cruse, H., Schmitz, J., Braun, U., Schweins, A. (1993b): Control of body height in a stick insect walking on a treadwheel. J. exp. Biol. 181, 141-155

Cruse, H., Silva Saavedra, M. (1996): Curve walking in crayfish. J. exp. Biol. (in press)

Dean, J. (1990): Coding proprioceptive information to control movement to a target: simulation with a simple neural network. Biol. Cybern. 63, 115-120

Dean, J. (1991a): A model of leg coordination in the stick insect, Carausius morosus. I. A geometrical consideration of contralateral and ipsilateral coordination mechanisms between two adjacent legs. Biol. Cybern. 64, 393-402

Dean, J. (1991b): A model of leg coordination in the stick insect, Carausius morosus. II. Description of the kinematic model and simulation of normal step pattern. Biol. Cybern. 64, 403-411

Dean, J. (1992a): A model of leg coordination in the stick insect, Carausius morosus. III. Responses to perturbations of normal coordination. Biol. Cybern. 66, 335-343

Dean, J. (1992b): A model of leg coordination in the stick insect, Carausius morosus. IV. Comparison of different forms of coordinating mechanisms. Biol. Cybern. 66, 345-355

Dean, J., Wendler, G. (1982): Stick insects walking on a wheel: perturbations induced by obstruction of leg protraction. J. Comp. Physiol. 148, 195-207

Espenschied, K.S., Quinn, R.D., Chiel, H.J., Beer, R.D. (1993): Leg coordination mechanisms in the stick insect applied to hexapod robot locomotion. Adaptive Behavior 1, 455-468

Espenschied, K.S., Quinn, R.D., Chiel, H.J. and Beer, R.D. (1994): Biologically-inspired hexapod robot control. Proceedings of the 5th International Symposium on Robotics and Manufacturing (ISRAM 94), Maui, August 14-18,1994.

Espenschied, K.S., Quinn, R.D., Beer, R.D. and Chiel, H.J. (1996): Biologically-based distributed control and local reflexes improve rough terrain locomotion in a hexapod robot, to appear in Robotics and Automation.

Ferrell, C. (1993): Robust agent control of an autonomous robot with many sensors and actuators. MS thesis, MIT, Cambridge.

Franceschini, N., Pichon, J.M., Blanes, C. (1992): From insect vision to robot vision. Phil. Trans. R. Soc. 337,283-294

Gibson, J.J. (1966): The senses considered as perceptual systems. Boston, Houghton Mufflin 1966

Graham, D. (1985): Pattern and control of walking in insects. Advances in Insect Physiology, 18, 31-140

Holst, E. v. (1943): Über relative Koordination bei Arthropoden. Pflügers Archiv 246, 847-865

Houk, J.C., Keifer, J., Barto, A.G. (1993): Distributed motor commands in the limb premotor network. Trends in Neurosciences 16, 27-33

Kristan, W.B., Jr., Lockery, S.R., Wittenberg, G., Brody, D. (1992): Making behavioral choices with interneurones in a distributed system. In: Neurobiology of Motor Programme Selection. J. Kien, C.R. McCrohan, W. Winlow (eds.). Pergamon Press pp. 170-200

Maes, P. (1991): A bottom-up mechanism for behavior selection in an artificial creature. In: From animals to animats. J.A. Meyer, S.W. Wilson (eds.). pp. 238-246. MIT Press, Cambridge, MA

Minsky, M. (1985): The Society of Mind. Simon and Schuster, New York

Pfeiffer, F., Eltze, J., Weidemann, H.J. (1994): The TUM walking machine. Proceedings of the 5th Intern. Symp. on Robotics and Manufacturing. M. Jamashidi, J. Yuh, Ch. Nguyen, R. Lumia (eds.), Vol. 2, pp. 167-174. ASME Press, New York

Schmitz, J., Bartling, C., Brunn, D.E., Cruse, H., Dean, J., Kindermann, T., Schumm, M., Wagner, H. (1995): Adaptive properties of "hard-wired" neuronal systems. Verh. Dtsch. Zool. Ges. 88.2, 165-179

Schmitz, J., Haßfeld, G. (1989): The treading-on-tarsus reflex in stick insects: phase dependence and modifications of the motor output during walking. J. exp. Biol. 143, 373-388

Steinkühler, U., Beyn, W.J., Cruse, H. (1995) A simplified MMC model for the control of an arm with redundant degrees of freedom. Neural Processing Letters 2, 11-15

Weidemann, H.J., Eltze, J., Pfeiffer, F. (1993): Leg design based on biological principles. IEEE Int. Conf. Robotics and Automation, IEEE Computer Soc. 1993, 352-358

Wendler, G. (1964): Laufen und Stehen der Stabheuschrecke: Sinnesborsten in den Beingelenken als Glieder von Regelkreisen. Z. vergl. Physiol.48, 198-250

Orientation Behavior Using Registered Topographic Maps

Cynthia Ferrell *
Massachusetts Institute of Technology
Artificial Intelligence Laboratory
545 Technology Square, Room 819
Cambridge, MA 02139 USA
voice: (617) 253-7007
fax: (617) 253-0039
email: ferrell@ai.mit.edu

Abstract

The ability to orient toward visual, auditory, or tactile stimuli is an important skill for systems intended to interact with and explore their environment. In the brain of mammalian vertebrates, the Superior Colliculus is specialized for integrating multi-modal sensory information, and for using this information to orient the animal to the source sensory stimuli, such as noisy, moving objects. Within the Superior Colliculus, this ability appears to be implemented using layers of registered, multi-modal, topographic maps. Inspired by the structure, function, and plasticity of the Superior Colliculus, we are in the process of implementing multi-modal orientation behaviors on our humanoid robot using registered topographic maps.

In this paper, we explore integrating visual motion and oculomotor maps to study experience-based map registration mechanisms. Continuing work includes incorporating self-organizing feature maps, including more sensory modalities such as auditory and somatorsensory maps, and extending the motor repertoire to include the neck and body degrees of freedom for full-body orientation.

1 Introduction

The ability to orient to sensory stimuli is an important skill for autonomous agents that operate in complex, dynamic environments. In animals, orientation behavior serves to direct the the animal's eyes, ears, nose, and other sensory organs to the source of sensory stimulation. By doing so, the animal is poised to assess and explore the nature of the stimulus with complementary sensory systems, which in turn affects and guides ensuing behavior. Hence, orientation behavior is performed frequently and repeatedly by agents that are tightly coupled with their environment, where perception guides action and behavior assists in more effective perception.

Certainly, orientation behavior is a basic skill we would like to implement on Cog, our humanoid robot (Brooks & Stein 1994). We would like Cog to perform a variety of tasks, many of which fall under two broad behavioral themes: exploratory behavior and social skills. For example, orienting the body to an object of interest assists manipulation tasks by putting the object where it is most accessible to sensory and motor systems. Eventually the work presented in this paper will be integrated with the ability to reach for visual targets (Marjanovic, Scassellati, & Williamson 1996). The same is true for social skills where the robot should position itself so that the person is easy to interact with.

Our approach to implementing orientation behavior on Cog is heavily inspired by relevant work in neuroscience (Stein & Meredith 1993), (Brainard & Knudsen 1993). In the brain of mammalian vertebrates, the Superior Colliculus is an organ specialized for producing orientation behavior. In non-mammalian vertebrates (birds, amphibians, etc.), the optic tectum is the analogous organ. The structure of the Superior Colliculus is characterized by layers of topographically organized maps. Collectively, they represent the sensorimotor space of the animal in ego-centered coordinates. These maps are interconnected and interact is such a way that the animal performs orientation movements in response to sensory stimuli.

Topographically organized maps have been discovered throughout the brain of mammalian vertebrates. In addition to the Superior Colliculus, they have been identified in various perceptual areas of the neocortex (the visual, auditory and somatosensory corticies, for instance). It is widely recognized that the organization of these maps are plastic and can be shaped through experience.

*Support for this research was provided by a MURI grant under the Office of Naval Research contract N00014-95-1-0600.

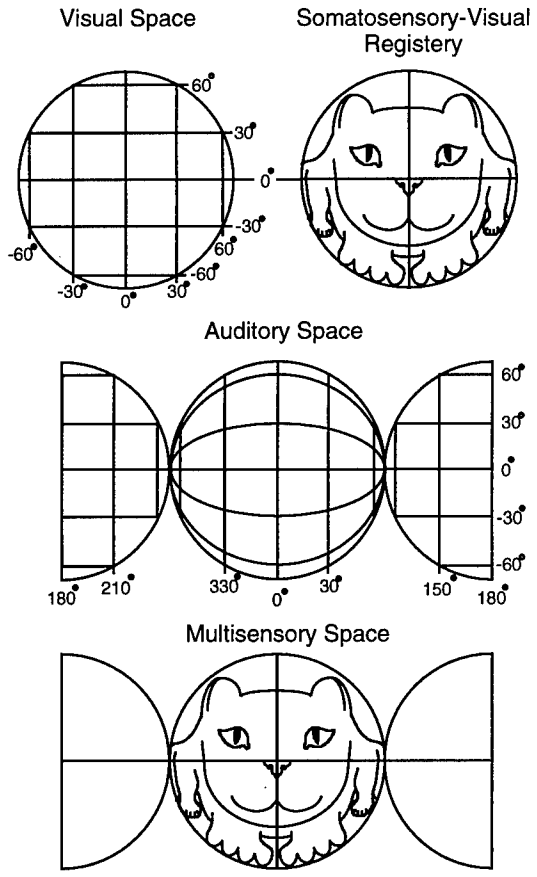


Figure 1: The Superior Colliculus is organized into layers of topographic maps. A variety of sensory maps, motor maps, and multi-modal maps have been discovered. These maps are registered with one another to share a common multisensory coordinate system. This figure illustrates registered visual, auditory, and somatorsensory spatial representations. Adapted from (Stein & Meredith 1993).

Subsequently, cortical maps have garnered a lot of attention, and a variety of work has explored the phenomena of self-organizing feature maps, (Kohonen 1982), (Ritter & Schulten 1988), (Obermayer, Ritter, & Schulten 1990).

Through implementing something like the Superior Colliculus on Cog, this paper explores how topographically organized maps develop and interact to produce a unified observable behavior. There are a variety of topics this paper explores in relation to this endeavor. From a behavioral perspective, we explore how dynamic spatio-temporal representations of sensory and motor space can be used to integrate multi-modal information and produce coherent behavior. From a developmental perspective, we investigate experience dependent mechanisms by which these maps self-organize and interconnect with one another. Given that behavioral experience affects both the connectivity within and between maps, and that the current state of connectivity dictates behavioral performance, we are dealing with a coupled system where the dynamics of forming connections affects and is effected by behavioral performance. Eventually, we would like to explore the dynamics of development where we expect to observe different developmental time scales of map self-organization and inter-map integration as the orientation behavior performance improves.

The rest of this paper is organized as follows. First we will briefly cover the organization, structure, and function of the Superior Colliculus, as our implementation is strongly inspired by what is understood about this organ. Next, because the physical architecture of the robot and the computer places heavy constraints on our implementation, we describe Cog, the experimental platform used in our experiments. After this, we present the state of our implementation at the time this paper was written, as well as extensions currently under development. Finally, we present tests and results of our system to date, and conclude with a brief description of ongoing work and future directions.

2 The Superior Colliculus

The Superior Colliculus is a midbrain structure composed of seven laminar layers. The deep layers are those believed to play a role in orientation behavior. Its physiology reflects its primary role as integrating different modalities to evoke motor responses. For example, among its many different afferent and efferent connections, it receives inputs from several sensory modalities such as the visual, auditory, and somatosensory corticies, and sends outputs to brain stem and spinal cord.

An important function of the Superior Colliculus is to pool sensory inputs from different modalities and redirect the corresponding sensory organs (eyes, ears, nose) to fixate on the source of the signal. Through the convergence of sensory inputs, the Superior Colliculus gives different sensory modalities access to the same motor circuitry so that any sensory modality can be used to direct the other sensory modalities to the source of the stimulus. For instance, by doing so, the animal can hear a sound emanating from an object outside its visual range (perhaps coming from behind the animal) and quickly turn its head and eyes to foveate on whatever is making the noise.

2.1 Organization of the Superior Colliculus

Localized regions of the Superior Colliculus (or Optic Tectum) consist of neurons with receptive fields that form topologically organized maps. A variety of topological maps have been identified in several species (cats, monkeys, owls, electric eels, and frogs to name a few). Each map corresponds to either a single modality or a combination of modalities. The modalities represented by the maps varies between species, depending on those

sensory or motor systems used in orientation behavior.

In the cat, there are visuotopic maps representing motion in visual space, somatotopic maps yielding a body representation of tactile inputs, and spatiotopic maps of auditory space encoding inter-aural time differences (ITD) and inter-aural intensity differences. Hence, a sensory stimulus originating from a given direction will elicit activity in the corresponding region of the appropriate sensory map. For example, an object moving in the visual field causes the corresponding region in the visuotopic map to become active. There are also motor movement maps consisting of pre-motor neurons whose movement fields are topologically organized. In the cat, these exist for the eyes, head, neck, body, ears. For example, stimulating a specific region in the oculomotor map elicits a movement to fixate on the corresponding area.

2.2 The Role of Map Registration

These multi-modal maps overlap and are aligned with each other so that they share a common multisensory spatial coordinate system. The maps are said to be *registered* with one another when this is the case. Arranging multi-modal information into a common representational framework within each map and aligning them allows the information to interact and influence each other. There are several advantages to this organizational strategy. First, it is an economical way of specifying the location of peripheral stimuli, and for organizing and activating the motor program required to orient towards it; thereby allowing any sensory modality to orient the other sensory organs to the source of stimulation. Second, it supports enhancement of simultaneous sensory cues. Stimuli that occur in the same place at the same time are likely to be interrelated by common causality. For instance, a bird rustling in the bushes will provide both visual motion and auditory cues. During enhancement, certain combinations of meaningful stimulus become more salient because their neuronal responses are spatio-temporally related. Once the multi-modal maps are aligned, neuronal enhancement (or depression) is a function of the temporal and spatial relationships of neural activity among the maps.

2.3 Development and Experience Dependent Plasticity

During development, the organization of the topographic maps is plastic. For each map, its representation of space is use dependent. In monkeys, it has been found that the organization of somatosensory maps for the hand can be changed by varying the amount and location of stimulation(s). Experiments have shown that the size of the map region corresponding to a particular cutaneous region on the hand is correlated to how much stimulation that part of the hand receives over time. Furthermore, adjacent regions of the map correspond to regions on the hand that are temporally adjacent (Stein & Meredith 1993). This phenomena has been seen in other perceptual areas of the cortex and is typical of self-organizing feature maps (SOFMs). A number of people have modeled this phenomena using neural networks (Kohonen 1982), (Bauer & Pawelzik 1992), (Durbin & Mitchison 1990).

It has also been found that the registration between maps is malleable over the developmental period. This phenomena has been studied in the inferior and superior colliculus of young barnyard owls, where the registration of the auditory map to the visual map shifts according to experience. (Brainard & Knudsen 1993), (Brainard & Knudsen 1995) found that the visual map is used to train the auditory map so that the auditory map shares the same coordinates as the visual map. Even if the visual map is artificially shifted by mounting distortion spectacles on the owls, the auditory map also shifts to keep in register with the visual map.

3 Cog: The Experimental Platform

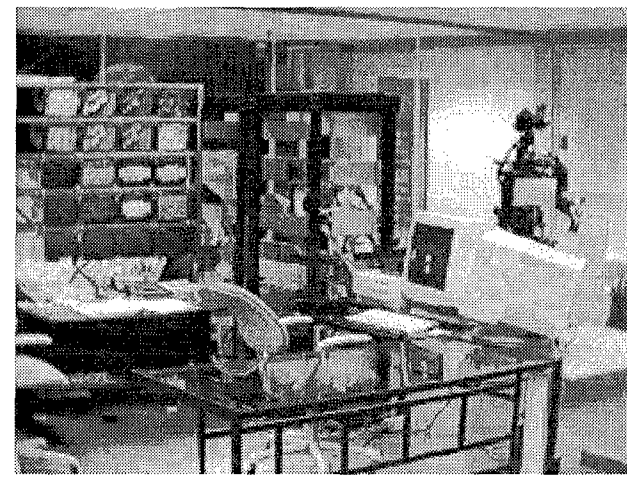


Figure 2: Cog is the humanoid robot used in our experiments, shown on the right. The "brains" of cog is a MIMD computer shown in the center of this image. The contents of DPRAMs (images, processed images, maps, etc.) can be displayed on a bank of twenty displays shown on the left. The Macintosh Quadra is the front end to the MIMD computer.

This section presents an overview of Cog, the humanoid robot used in our experiments. We briefly describe the implementation of the robotic platform, the perceptual systems, the computational system, and the software systems relevant to this paper. All the systems described below were designed and constructed by members of the Cog Project (see acknowledgements).

3.1 The Robotic Platform

A fundamental design goal for the robot was to make it anthropomorphic as possible so that 1) the robot could move in a human-like manner, and 2) encourage humans to interact with it in a natural way. The most important human characteristics to emulate are size, speed, and range of motion. Hence, Cog resembles a human from the waist up and is shown in figure 2.

Cog's body has six degrees of freedom (DOF): the waist bends side-to-side and front-to-back, the "spine" can twist, and the neck tilts side-to-side, front-to-back, and twists left-to-right. Mechanical stops on the body and neck give a human-like range of motion. In addition, each degree of freedom has current sensing in the motor controller to provide some force feedback, temperature sensing to determine a longer term time-average of how hard the motors are working, and joint encoders to provide a proprioceptive sense.

Cog's head is equipped with a compact, binocular, active vision system. To maintain an anthropomorphic appearance, the "eyes" were mounted about 3 inches apart. To mimic human eye movements, each "eye" can rotate about a vertical axis (pan DOF) and a horizontal axis (tilt DOF). To approximate the range of motion of human eyes, mechanical stops were included on each eye to permit a 120° pan rotation and a 60° tilt rotation. In addition, each eye performs fine motor control and high-speed positioning so that we may emulate human visual behaviors.

3.2 The Perceptual Systems

To give the robot both a wide field of view and a high resolution foveal area, each eye consists of two black and white CCD cameras. We could have simplified our design by using a single camera per eye. However, by using two cameras per eye we have a much higher resolution fovea than the single camera eye. The lower camera of each eye gives Cog a wide peripheral field of view ($88.6°(V) \times 115.8°(H)$ FOV), and the upper camera of each eye gives Cog a high resolution fovea ($18.4°(V) \times 24.4°(H)$ FOV).

In addition to the visual system described above, Cog has several other perceptual systems under concurrent development. To date, we have developed an auditory system (Irie 1995), a vestibular system, and a variety of force resistive sensors to give Cog a tactile sense. These systems are in the process of being ported to the Cog platform.

3.3 The Computational Platform

This section summarizes the computational system we developed to meet Cog's requirements. First, the system must allow for real-time control of the robot since the robot operates in a dynamic environment full of people and objects we would like the robot to interact with. Furthermore, the system must be robust, scalable, concurrent, and support learning and development processes.

Cog's "brain" is a scalable MIMD computer consisting of up to 256 processor nodes. Currently the nodes are based on the Motorolla 68332, but Texas Instruments C32 DSP nodes are under development which will be responsible for the bulk of perceptual processing. Processors can communicate through eight ports to other processor nodes or to other parts of the video capture/display system. All components of the processing system communicate through dual-ported RAM (8K by 16 bits) connections, so altering the topology is relatively simple. During operation, the brain is a fixed topology network which can be changed manually and scaled by physically adding additional nodes and dual-ported RAM connections. The entire brain is connected through a serial line with a Macintosh Quadra, which is used for communication and input, but not for any actual processing. Each node also uses standard Motorola SPI (serial peripheral interface) to communicate with up to 16 motor controller boards.

The video capture/display system consists of custom designed frame grabbers and display boards. A frame grabber takes the NTSC signal from one of the eye cameras, digitizes the input, subsamples it to 128 by 128 pixels, and writes the resulting grayscale values to six ports. For this design, we have chosen to use only 128 by 128 grayscale in order to reduce the amount of data to be processed and to increase the speed of the visual processing. The frame grabbers operate at approximately 30 frames per second, and can write simultaneously to six processors. The display boards produce NTSC video output for display on a bank of 20 display monitors. Direct camera output and digitized output of the frame grabbers can also be routed directly to the monitor bank.

A network of special purpose motor controller boards mounted on the robot act as Cog's "spinal cord", connecting the robot's "brain" to the rest of the body. Each motor has a dedicated motor controller board that reads the encoder (and other sensors), performs servo calculations, and drives the motor. The motor controller boards have hardware that generates a 32KHz PWM waveform. The duty cycle is updated at 2KHz by an on-board MC6811E2 microcontroller. Currently the microcontroller implements a PID control law for position and velocity control. Position and velocity commands are sent to the motor controller boards from the MIMD computer described above.

3.4 The Software Environment

Each processor has an operating system; *L*, a compact, downwardly compatible version of Common Lisp that supports real-time multi-processing (Brooks 1994*a*); and *MARS*, which is a front end to *L* that supports com-

munication between multiple processes on a single processor as well as communication between processes running on separate processors (Brooks 1994*b*). *MARS*, like the *Behavior Language* (Brooks 1990), is a language for building networks of concurrently running processes. The processes can communicate either locally by passing messages over virtual wires, or globally through a process inspired by hormonal mechanisms. *MARS*, unlike the *Behavior Language*, supports on-line learning mechanisms by allowing the network morphology to change dynamically, i.e. spawning or killing processes or changing network connectivity during run-time.

4 A Developmental Approach to Orientation Behavior

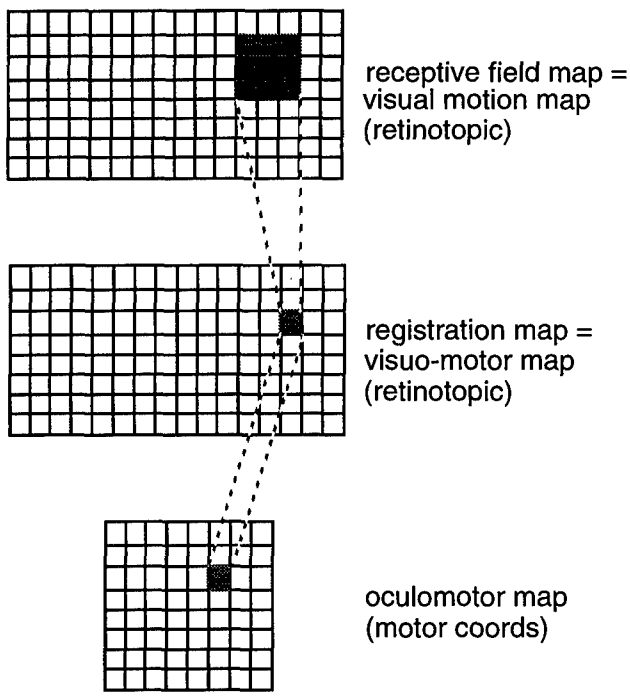


Figure 3: Example of layered topographic maps implemented on Cog. The multi-modal registration map acts to relay activity in the receptive field map (the visual motion map) to activate the oculomotor map such that a visual stimuli is foveated. See section 4.1 for further explanation.

Registered topographic maps form a substrate upon which multi-modal information can be integrated to produce coherent behavior. How are these topographic maps formed? How do they become registered with one another? How is the organization of the ensemble guided by experience?

4.1 The Framework

In our framework, a map is a two dimensional array of elements where each element corresponds to a site in the map. The maps are arranged into interconnected layers, where a given map can be interfaced to more than one map. Each connection is uni-directional, so recurrent connections between maps require both a feedforward connection and a feedback connection. The activity level of sites on one map is passed to another map thorough these connections, hence the input to a given map is a function of the spatio-temporal activity of the maps feeding into it and the connectivity between these maps. Currently, all connections have equal weights, although this could change in the future. The output of a given map is its spatio-temporal activity pattern. What this pattern of activity represents depends upon the map: if it is a visuotopic map, it could represent motion coming from a particular direction in the visual field; if it is an oculomotor map, it could encode a motor command to move the eyes, and so forth.

The smallest map ensemble capable of producing an observable behavior consists of a sensory input map, a motor output map, and an established set of connections between them. The input map could have a fairly rigid structure consisting simply of time-differenced intensity images. Because visual information already contains a spatial component, this simple map is topographic without any additional tuning. The motor map could also be fixed where a given site on the map corresponds to a given motor command. If the motor commands vary linearly with motor space, for instance, this map is also topographically organized. Assuming the cameras are motionless, a moving object occupies a localized region in the visual field, and correspondingly causes a localized intensity difference (an active region) in the time-differenced image map. If there exists connections from this region of the time-difference map to the appropriate region of the oculomotor map, then a motion stimulus in the visual field activates the corresponding region of the time-difference map, which in turn excites the connected region of the oculomotor map, which evokes the necessary camera motion to foveate the stimuli.

4.2 Developmental Mechanisms

Plasticity can be introduced into the simple system above in two ways: 1) the map organization could change so that a given map site could correspond to different locations in space. 2) The connections between maps could change so that a given site could change which site(s) it connects to on the other map.

In animals, as described in section 2.3, the organization of the maps and the registration between maps is tuned during the critical period of development. Several mechanisms and models have been proposed to account

for this organizational process. The mechanisms we use for map organization and alignment on Cog are inspired by similar mechanisms (Kohonen 1982). However, different combinations of mechanisms are used depending on what is being learned: i.e. tuning the organization within a map, registering different sensory maps, or registering sensory maps and motor maps.

A variety of mechanisms determine how map connections are established. Guided by sensori-motor experience, these mechanisms govern how connections are modified to improve behavioral performance.

- *Competition*: There is competition between concurrently active sites where only the most active site is modified per trial. In our system, the most active is currently approximated as the centroid of activity of the active region. Furthermore, each site of a given map can only form a limited number of connections to the other map. So, candidate map sites compete to determine those that can connect to a given site on the other map.

- *Locality, neighborhood influences*: The neighboring sites around the most active site are also updated each trial. The amount a neighboring site is adjusted decays with distance from the maximally active site. This mechanism penalizes long connections and encourages topographic organization. The size of the neighborhood can vary over time. Typically, it starts off fairly large until the map displays some rough topographic organization, then it decreases as the map undergoes fine tuning adjustments.

- *Error correction*: It is not sufficient that the maps are topographically organized and aligned – they must be organized and interfaced so that the agent performs well in its environment. For tight feedback loop sensorimotor tasks (such as saccading to a visual stimulus), an appropriate error signal is very important and useful for tuning the behavior of the system. Naturally, the error signal must be a good measure of performance and obtainable at a fast enough rate to enable on-line learning. Connections are modified to reduce the discrepancy between current performance and desired performance. The magnitude of the correction is proportional to the size of the error on that trial.

- *Correlated temporal activity*: Hebbian mechanisms are often used for self-organizing processes. By strengthening connections between simultaneously active sites, they are useful for relating information between different sensory maps.

- *Learning rate*: The magnitude of the adjustment for each trial is also proportional to the learning rate. The learning rate can vary over time, where it starts of relatively large for course tuning, and then decreases for finer adjustments.

4.3 A Simple Behavior

In this section we look at an example to see how these mechanisms are applied to forming and organizing these multiple maps to perform a task. A simple orientation task is the ability to saccade to noisy, moving stimuli (clapping hands, shaking a rattle, etc.). We say that a good saccade centers the stimulus in the fovea camera's field of view, whether the stimulus is seen in the wide field of view or the fovea field of view. We assume that the system favors information from the foveal view because it is of higher resolution than the peripheral view and thereby can be used to perform a more accurate saccade.

Experience dependent plasticity could play a role in several ways. It could be used to guide the representational organization of the auditory and visual motion maps, guide the registration between the auditory and visual motion maps, or guide the registration of the the sensory maps to the oculomotor map. Below, these three types of organization are explored in turn:

- Self organizing feature maps
- Registration of sensory maps
- Registration of sensory and motor maps.

Each mapping process can be viewed as learning a multi-modal map that registers the information from two other modality maps. We call the multi-modal map the *registration map*, and the other maps could be either sensory maps, motor maps, or both. One of the modality maps provides the rough spatial organization of the multi-modal map. We call this map the *receptive field map*. Typically it has a topographic representation of space. Often a retinotopic map is used, for instance. The second modality map, which may or may not be topographic, is registered to the first map through the multi-modal map. This process is illustrated in figure 3. Note that each site of a modality map connects to only one site on the registration map, but the same site on the registration map could connect to mulitple sites on the modality maps.

4.3.1 Self Organizing Feature Maps

One example of a self organizing feature map is learning the visual motion map for the peripheral field of view. In this case, the receptive field map is a time-difference map of consecutive intensity images (in retinotopic coordinates) and the registration map is the visual motion map. Initially, the receptive field map contains broad, overlapping receptive fields for the corresponding sites of the registration map. Those connections that are spatio-temporally correlated are strengthened over time,

whereas those that are not are weakened and eventually die off. Hence, the primary developmental mechanisms used for this case are competition, neighborhood updates, and hebbian learning.

Recall that the organization of the registration map will reflect how the map is used. The resulting visual motion map should represent that motion in peripheral view that is relevant to behavior. This is not necessarily a direct mapping from the intensity time-difference map. For example, if motion is present in the fovea region, then the system should favor this information over the information coming from the peripheral view. Over time, we would not expect to see the center $20° \times 20°$ of the peripheral field of view represented in the peripheral motion map because this information is not used to perform saccades.

4.3.2 Registration of sensory-sensory maps

An example of aligning sensory maps is registering the auditory map to the visual motion map. In this case, the receptive field map is the visual motion map (in retinotopic coordinates), the registration map is a visuo-auditory map (also in retinotopic coordinates), and the third map is the auditory ITD map. The auditory ITD map could be tonotopically organized, instead of topographically organized, since auditory signals do not contain inherent spatial information. Initially, the visual map and the auditory map are interfaced to the registration map. The registration map contains broad, overlapping receptive fields. Registration of the auditory map with the visual map entails mapping the correct ITD values to the corresponding regions in the registration map that are in turn connected to the visual map where specific locations in visual coordinates are represented by spatial location. During the developmental process, those visual and auditory signals that are spatio-temporally correlated are strengthened, and those that are not eventually die off. Hence there is a significant amount of weeding out of inappropriate connections. Over time, the ITD receptive fields in the registration map become refined and properly located in retinotopic coordinates. Here, the primary developmental mechanisms are competition, neighborhood updates, and hebbian learning.

4.3.3 Registration of Sensory-Motor Maps

An example of aligning sensor and motor maps is registering the oculomotor map with the visual motion map. In this case, the receptive field map is the visual motion map (in retinotopic coordinates), the registration map is a visuo-motor map (also in retinotopic coordinates), and the third map is the oculomotor map (in eye motor coordinates). Regions in the motor map correspond to motor movements that could foveate a stimulus. Initially the visual map and the oculomotor map are connected to the registration map with broad, overlapping receptive fields. When the motion map is stimulated and the site of maximal activity is determined (typically the centroid of the stimulated region), the corresponding region of the oculomotor map is stimulated. The site of maximal response of the motor map is taken as the motor command, and the corresponding motor movement is evoked. This movement orients the eye to the stimulus. Once oriented, the motion stimulus stimulates a different region in the visual motion map. The visual error is computed as the difference from centroid of motion to the center of the field of view. This error is used to update the connections responsible for the orientation movement to reduce the error in the future. Hence, the primary developmental mechanisms are competition, neighborhood updates, and error correction.

5 Architectural Organization

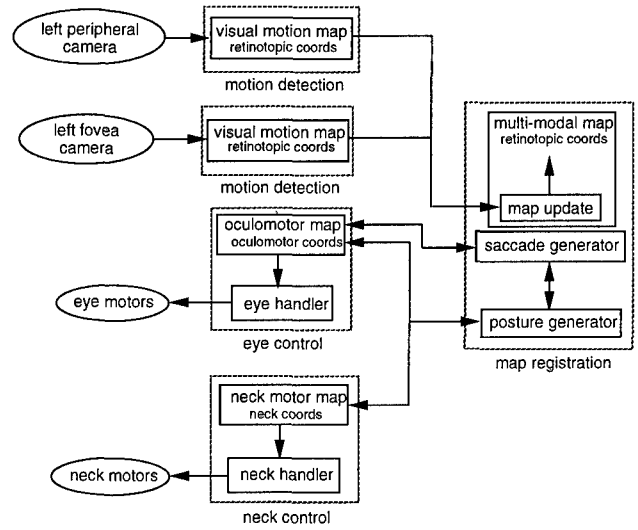


Figure 4: This diagram shows how the multi-modal topographic maps are arranged on Cog's computational hardware. Currently five processors are used: two visual processors, two motor control processors, and one processor which performs the developmental mechanisms. See text for further explanation.

To date, the sensory-motor map registration task has been implemented on Cog's hardware and is shown in figure 4. The diagram shows how the processes are arranged on Cog's MIMD computer. Currently, five processing nodes are used:

- *Peripheral motion processor*: Contains the peripheral visual motion map. It computes the difference between consecutive left peripheral camera images at 15 frames/s. It also determines the most active site (the centroid of motion) and a visual error signal.

- *Fovea motion processor*: Contains the fovea visual motion map. It computes the difference between consecutive left fovea camera images at 15 frames/s. It also determines the most active site (the centroid of motion) and a visual error signal.

- *Registration processor*: Contains the visuo-motor map and carries out the developmental process. It receives motion information from the vision processor and determines which motion information to use. If fovea motion is present, it ignores the information from the peripheral camera. It also translates the most active site on the visual map to the region of activity on the motor map, and passes this information to the oculomotor processor. After the motion is performed, it uses the error signal from the vision processors to update the registration map connections according to developmental mechanisms.

- *Oculomotor processor*: Contains the oculomotor map. Upon receiving the site of activity, it commands the motors to perform the movement. It also sends an "efferent copy" to the registration processor, so the registration processor can ignore visual motion information while the cameras are moving.

- *Neckmotor processor*: Contains the neckmotor map. It is commanded by the registration processor to move the neck around so the motion stimulus is seen from many different places in the visual field. Currently, the neck is primarily used for the training processes. However, soon it will incorporated into the orientation behavior.

6 Tests

To date we have run experiments to test whether the implementation we have described learns the registration between the retinotopic visual motion map and the oculomotor map. A sampling of our results are shown in figures 5, 6, and 7.

So far, experiments have been performed using the left eye only. The system will be extended to handle both eyes when stereopsis and vergence capabilities are implemented. Motion information from both eyes will be fused and used to excite the visuotopic motion map. Conflicts between the eyes will be resolved during this fusion stage. The simplest approach would be to resolve conflicts via a dominant eye mechanism. Another method could involve exciting the visuotopic map with the stonger of the two excitations coming from each eye. These two methods along with other possibilities need to be explored. Most likely, a combination of methods will be implemented.

To learn the registration between the peripheral motion map with the oculomotor map, we trained Cog over a number of trials while it looked at a continuously moving stimulus. At the beginning of each trial, the robot

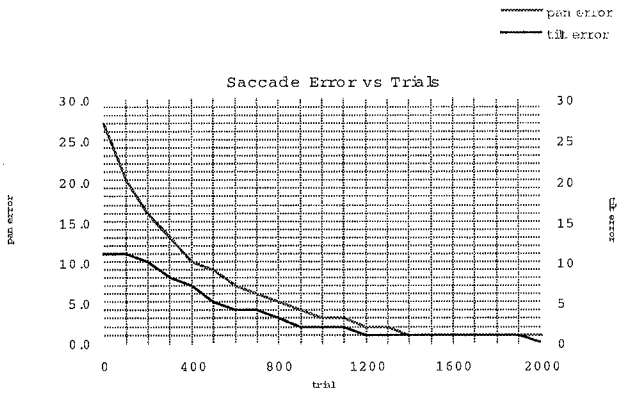


Figure 5: Registration data for aligning the visual motion map and the oculomotor map. The data is derived from the registration map, converting sites in visuotopic coordinates to activated sites in the motor movement map (pan and tilt) required to foveate the stimulus. Initially the maping is random, with a neighborhood radius size of 2 and a learning rate of .25. The neighborhood size remained fixed for all trials in this experiment. At $trial = 0$, the average error over the 20×20 region was approximately $(11°, 26°)$ for pan and tilt DOFs respectively. By $trial = 400$, the average error is reduced to $(10°, 7°)$; it is the slowest to converge to an average error $\leq (1°, 1°)$ of the three experiments shown in this section. By the 1400th trial, the average error was close to 1° for pan and tilt DOFs.

changes its posture (centers its eyes and moves its neck to a random location). This places the motion stimulus in a different location in its visual field. Currently Cog explores the center $20° \times 20°$ of the peripheral visual field, which corresponds to a 20×20 region of the registration map. The robot uses the visual information to stimulate the oculomotor map and perform the saccade. The visual error is then acquired, and the registration between the maps is updated according to the rule:

$$\Delta m(x, y) = \rho \times \epsilon(x, y) \times N(x, y) \qquad (1)$$

where:

- $m(x, y)$ is the value of site (x, y) of registration map m. Recall that this value represents the connection from the visual motion map site to the corresponding oculomotor map site. The learning process involves updating these inter-map connections.

- (x^*, y^*) is the site of maximal activity of the motion map. For this application, it corresponds to the site of maximal activity of the registration map as well.

- ρ is the learning rate.

- $\epsilon(x, y) = target(x, y) - m(x, y)$. It is an error distance measure between the motion map site $m(x, y)$ and

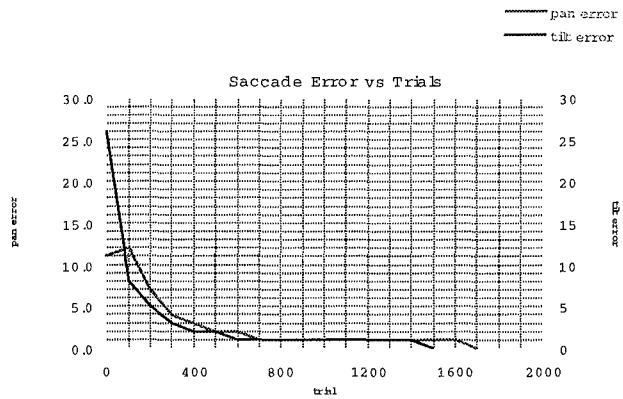


Figure 6: Same experiment as shown in figure 5, except the neighborhood size is manually decreased over time. Initially the maping is random, with a neighborhood radius size of 4 and a learning rate of .25. By $trial = 400$, the average error is reduced to $(3°, 2°)$ and the neighborhood size was set equal to 2. This experiment is the second fastest to converge to an average error $\leq (1°, 1°)$ of the three shown in this section. By the 800th trial, the average error was close to 1° for the pan and tilt DOFs.

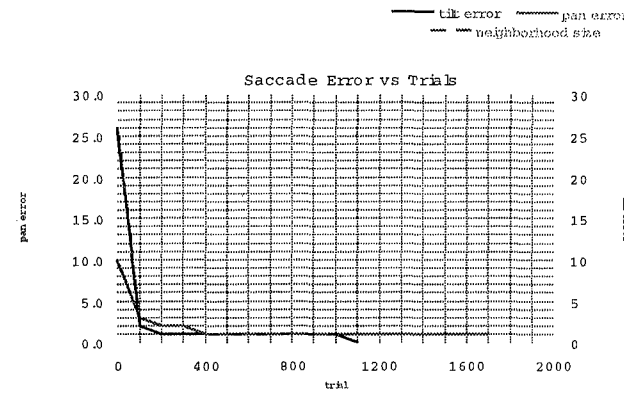


Figure 7: Same experiment as shown in figure 5 and figure 6, except the neighborhood size is automatically decreased over time. Each trial, the neighborhood size is set equal to the larger value of the average error measures (pan or tilt). Initially the maping is random, with a neighborhood size of 26 and a learning rate of .25. By $trial = 100$, the average error is reduced to $(3°, 2°)$, and by $trial = 400$ it is reduced to $(1°, 1°)$. Of the three experiments shown, it is the fastest to converge to an average error $\leq (1°, 1°)$.

the target site. This measurement is made after the saccade motion finishes. Note that $target(x, y)$ is the center of the field of view for the saccade learning task.

- r is the neighborhood radius.
- $N(x, y) = f(1 - \frac{|(x^*, y^*) - (x, y)|}{r})$. It is the neighborhood update function that decays linearly with distance from the site of maximal activity (x^*, y^*). Threshold function, f, sets the result equal to zero if its argument is negative. So, for site locations outside radius r of (x^*, y^*) , $N(x^*, y^*) = 0$.

7 Continued Work

Currently, we are extending these tests to include the full visual field, and continuing our experiments with dynamically varying neighborhood size and learning rate parameters. Soon we will explore the self organizing properties of the representation of visual information by implementing a SOFM for visual motion. We will also begin efforts to register the visual motion with an auditory ITD map, as well as investigate the dynamics of development when self-organization and registration mechanisms are run simultaneously. We expect to see evidence for different developmental time scales as the robot learns the orientation task.

With the above work in place, we will extend the system to include the neck and body degrees of freedom so that the robot can perform full body orientation behavior. This will complicate the current task by adding additional degrees of freedom that must complement each other. We will continue to investigate the issue of developmental time scales since more complicated behaviors will have to develop incrementally and bootstrap off of existing behaviors. We would like to integrate the full orientation behavior with reaching and manipulation tasks currently under parallel development by other members of the group (Marjanovic et al. 1996).

8 Conclusions

This paper describes an implementation of orientation behavior on Cog using registered topographic maps. We have presented biological evidence that this is an effective method for orienting to multi-modal stimuli in animals. We have also presented a series of mechanisms and methods for developing this behavior on Cog over time. This biologically inspired framework gives us the opportunity to explore several interesting issues. It allows us to investigate using dynamic spatio-temporal representations of sensory-motor space to integrate multi-modal information and produce a unified behavior. It also allows us to investigate the dynamics of development using mechanisms of experience dependent plasticity. Ongoing work is promising.

Acknowledgements

A number of people have made this work and the Cog Project possible. Among those who have contributed to the ongoing development of Cog are (in alphabetical order): Mike Binnard, Rod Brooks, Robert Irie,

Eleni Kapogannis, Matt Marjanovic, Yoky Matsuoka, Brian Scasselatti, Nick Shectman, Rene Schaad, and Matt Williamson.

References

Bauer, H. & Pawelzik, K. (1992), 'Quantifying the Neighborhood Preservation of Self-Organizing Feature Maps', *IEEE Transactions on Neural Networks* **3**(4), 570–579.

Brainard, M. & Knudsen, E. (1993), 'Experience-dependent Plasticity in the Inferior Colliculus: A Site for Visual Calibration of the Neural Representation of Auditory Space in the Barn Owl', *The Journal of Neuroscience* **13**(11), 4589–4608.

Brainard, M. & Knudsen, E. (1995), 'Dynamics of Visually Guided Auditory Plasticity in the Optic Tectum of the Barn Owl', *Journal of Neurophysiology* **73**(2), 595–614.

Brooks, R. (1990), AIM 1227: The Behavior Language User's Guide, Technical report, MIT Artificial Intelligence Lab Internal Document.

Brooks, R. (1994*a*), L, Technical report, IS Robotics Internal Document.

Brooks, R. (1994*b*), MARS, Technical report, IS Robotics Internal Document.

Brooks, R. & Stein, L. A. (1994), 'Building Brains for Bodies', *Autonomous Robots* **1:1**, 7–25.

Durbin, R. & Mitchison, G. (1990), 'A Dimension Reduction Framework for Understanding Cortical Maps', *Nature* **343**(6259), 644–647.

Irie, R. (1995), Robust Sound Localization: an Application of an Auditory System for a Humanoid Robot, Master's thesis, MIT.

Kohonen, T. (1982), 'Self-Organized Formation of Topologically Correct Feature Maps', *Biological Cybernetics* **43**, 59–69.

Marjanovic, M., Scassellati, B., & Williamson, M. (1996), Self-Taught Visually-Guided Pointing for a Humanoid Robot, *in* 'Proceedings of the 4th Intl. Conference on Simulation of Adatpive Behavior', Cape Cod, MA.

Obermayer, K., Ritter, H., & Schulten, K. (1990), 'A Principle for the Formation of the Spatial Structure of Cortical Feature Maps', *Proceedings of the National Academy of Science USA* **87**, 8345–8349.

Ritter, H. & Schulten, K. (1988), Kohonen's Self-Organizing Maps: Exploring their Computational Capabilites, *in* 'Proceedings of the IEEE International Conference on Neural Networks', Vol. 1, San Diego, CA, pp. 109–116.

Stein, B. & Meredith, M. (1993), *The Merging of the Senses*, A Bradford Book, Cambridge, MA.

Locating Odor Sources in Turbulence with a Lobster Inspired Robot

Frank Grasso[1], Thomas Consi[2], David Mountain[3] and Jelle Atema[1]

[1]Boston University Marine Program
Marine Biological Laboratory
Woods Hole, MA 02543
fgrasso@hoh.mbl.edu
atema@bio.bu.edu

[2]MIT Sea Grant
Autonomous Underwater Vehicles
Laboratory
Cambridge, MA 02139
consi@mit.edu

[3]Biomedical Engineering
Boston University
Boston, MA 02215
dcm@buenga.bu.edu

Abstract

Our lobster bio-mimic, "Robolobster" is designed to test chemical orientation strategies that real lobsters and other animals might use to locate odor sources such as food and mates. The size, speed and maneuverability of Robolobster compare to those of the real animal but no effort is made to model lobster biomechanics. Instead, our studies emphasize chemosensory processing at the temporal scale of lobsters navigating to odor sources under turbulent flow conditions. Robotic explorations of "standard" (i.e. statistically reproducible) turbulent odor plumes using simple orientation strategies have revealed two distinct plume regions: the "proximal jet", where the orientation task is relatively easy and the "distal patch field" where simple algorithms have proven to be inadequate. Physical investigations of odor dispersal in turbulent plumes demonstrate that in the distal patch field a simple gradient ascent mechanism cannot reliably guide a robot to the source. This paper outlines our initial results and suggests alternative strategies the (Robo-) lobster might use in the distal patch field to navigate to the source.

1. Introduction

Marine animals that use chemical senses to locate food and mates face serious challenges when determining the direction of a chemical source. Common sense would suggest that an animal evaluates its progress towards an odor source by the increase or decrease in odor concentration along its path. However in many environments smooth concentration gradients radiating from the source seem to be the exception rather than the rule (Grasso, Basil and Atema.,1996 (in prep); Murlis *et.al.*,1992, Moore and Atema,1991). The patchy distribution of odor resulting from natural turbulence presents the odor-sampling animal with complex temporal patterns of odor concentration. Determining the source direction from these signals is a major computational challenge for the animal and the investigator.

Animals have developed a number of strategies for coping with the patchiness of real-world plumes. Some animals like tse-tse flies move through the odor plume in a sequence of straight high-speed flights. Before taking off these flies determine mean wind direction, then fly up-wind as long as odor is present, and down-wind when absent (Bursell, 1984,1987). Other animals like moths appear to use visually-guided, odor-gated rheo(=current)taxis. They move directly upwind (against the current) as long as they sense odor pulses (pulses in time = patches in space from the perspective of a moving animal). They cast across the local current during intervals without odor. By casting they can re-engage a lost plume (Mafra-Neto and Carde. 1994, Vickers and Baker. 1992). Blue crabs appear similar to moths (Zimmer-Faust *et. al.* 1995). The paths of orienting American lobsters do not resemble those of the Tsetse fly, Blue Crab or Moth. As the lobster proceeds the accuracy of its heading increases steadily as if its estimate of the source's direction improves with experience of that plume. For Blue crabs, moths and tsetse flies the accuracy of the heading declines as the animal approaches the source because the animal moves parallel to the one ideal line that proceeds down-wind/current from the source. This observation led us to propose that lobsters might use the temporal structure of the concentration pulses produced by turbulence to locate the source (Moore. Scholz and Atema, 1991, Basil and Atema, 1994, Basil, Grasso and Atema in prep.).

What might these temporal cues be? We identified and evaluated a number of purely chemical putative cues in studies of the physical characteristics of turbulent chemical dispersal under conditions where lobsters are known to locate odor sources (Moore and Atema, 1991; Grasso, Basil and Atema, 1996 in prep). The patchy distribution of odor under our experimental conditions indicated that the lobster would have to wait at each consecutive location in excess of 30 seconds to achieve a 50 % reliable average estimate of the concentration gradient. Behaviorally, orienting animals are never observed to pause this long while tracking an odor and selection pressure would favor the fastest solution to the orientation task. We identified two among the most reliable candidate cues: 1) tracking temporal parameters of pulse shape (slope, height etc.) and 2) cross correlation (CCF) of simultaneous paired sampling. This last cue is particularly interesting because these animals appear to

need both lateral antennules as the primary chemoreceptive organs when attempting to locate distant odor sources (Devine and Atema,1982). They can locate odor sources with a single functioning antennule but their efficiency is greatly compromised.

In parallel with our behavioral orientation studies we constructed "Robolobster" to test the hypothesis that the lobster could use the temporal patterns of odor concentration produced by turbulent dispersal processes. Robolobster allows us to test the practical utility of chemical guidance strategies in the absence of other forms of sensory information (rheotactic, visual, tactile).

Although neurophysiological studies of receptor cells in the lateral antennules showed them capable of coding features of odor pulse shape, particularly onset slope of concentration, (Gomez, Voigt and Atema 1995), we chose to explore first the possibility of guidance by simple bilateral signal differences. Using this alternative we could rely on both the physical plume studies and the behavioral studies for guidance in specifying algorithms. This report describes the results of some of these investigations. Some of the data presented here were published previously in a short report (Consi, Grasso, Mountain and Atema 1995).

2. The Robot Lobster

Our robot lobster is constructed to imitate the biological lobster in operational chemosensory (but not structural) detail. It has the approximate size, turning ability and speed of a lobster (TABLE I). It does not match the lobster in appendages or in mode of locomotion. In place of the lobster's eight walking legs, "Robolobster" has two motorized wheels and a castor.

In place of lateral antennules it has two conductivity sensors with which to sample chemical concentrations in its environment. In early experiments these sensors were two pairs of platinum wires separated by 5-7 cm (5 cm separating two 1 cm pairs of wires). In more recent experiments these sensors have been replaced by two pairs of vertical stainless steel rods with separations adjustable between 1 and 9 cm.

TABLE I. Robolobster Specifications

Length	**24 cm**
Width	**22 cm at wheel base**
Height	**Hull 13 cm (Sensors Extend 5 cm above the hull).**
Weight	**2.6 kg**
Displacement	**1.7 kg**
Maximum Speed	**40 cm/sec**
Turning Radius	**13 cm**

The robot is essentially a hollow waterproof Lexan cylinder mounted on a wheeled base with two independent gear motors (FIGURE 1). Turns are effected by differential

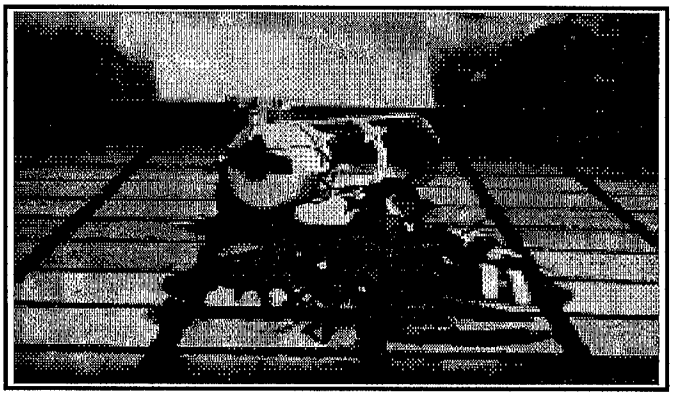


FIGURE 1. The Robolobster and biological counterpart on a 10 cm grid in the flume tank.

inputs to the motors. Robolobster can be operated with power from 16 on-board alkaline AA batteries or from a remote power source. For the results reported here the remote power supply was used.

The "brain" of Robolobster is a Tattle Tale Model 7 computer (Onset Corp., Motorola MC 68332 microprocessor) with 256K SRAM and a 20 MB Hard disk. This computer is programmed to process the sensor input and to control the rotation of the two wheel motors. An onboard Yaw Rate Gyro (Murata ENX-0011) is used for data analysis and path reconstruction but has not yet been incorporated into our orientation algorithms.

The computer is programmed in Think C^{tm}. We have developed a library of functions which execute basic motor commands and collect and log sensor data. From these building blocks, a wide range of chemical orientation algorithms can be constructed.

3. Orientation Studies with Robolobster

3.1 The Orientation Task

In our studies of Robolobster's ability to locate chemical signal sources, the orientation task is the same for robot and lobster. The physical conditions of flow, stimulus delivery and initial distance from the source that Robolobster faces replicate those of our experimental lobsters. Robolobster trials are run using a "standard turbulent jet plume" (see below) in the same flow-through flume as the lobster behavioral experiments. The principal difference between the experiments is the use of a salt plume in fresh water for the robot to simulate the food stimulus plume in salt water used with the lobster. Our studies of food odor dispersal were conducted using voltammetric probes scaled to the spatial and temporal resolution of the lobster chemoreceptor cell using dopamine as a chemical tracer (Moore and Atema 1991, Basil and Atema 1994, Grasso, Basil and Atema 1996 in prep). Similar studies of salt plumes did not identify significant dispersal differences between salt plumes in fresh water and

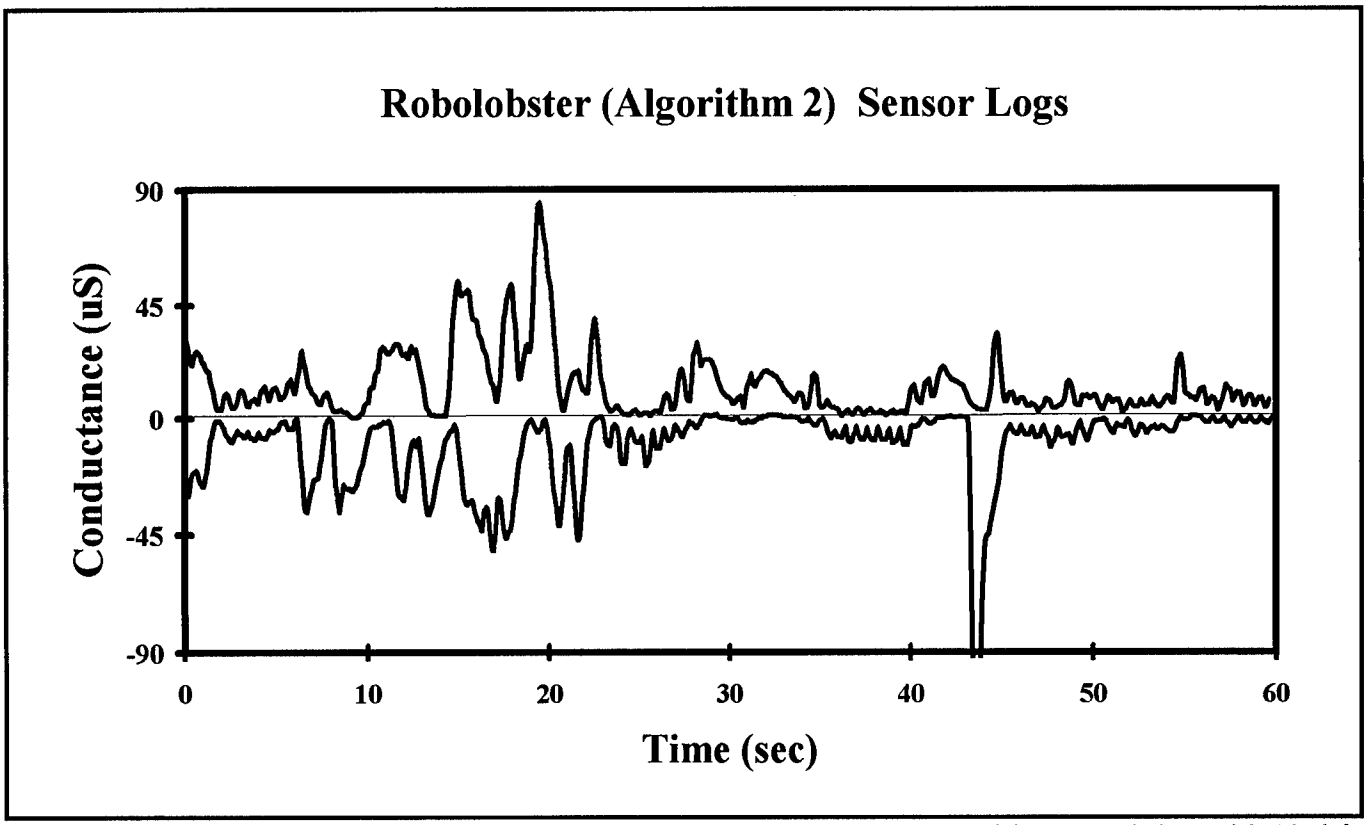


FIGURE 2. A typical pair of simultaneous concentration profiles collected by Robolobster conductivity sensors during a trial. The left sensor recording is inverted for ease of comparison with the right. Note the peaks/pulsed which mark the sensors passing through odor patches.

dopamine plumes in salt water (Dittmer, Grasso and Atema, 1995). During a typical orientation run, the real lobster holds its antennules perpendicular to the flume floor as it marches up the plume. Thus its receptors each sample a 2-D cross sectional plane parallel to the floor of the tank as the animal moves through it. So too with Robolobster, which has its two sensors positioned at a height comparable to that of the orienting lobster's lateral antennules.

The testing arena is a flow-through flume (366x90x36 cm) through which we run fresh water at a rate that gives a mean flow of 0.7 cm/sec. Collimators at the upstream end produce a unidirectional mean flow. Into this ambient flow we inject a plume of salt water at a rate which (given the same nozzle diameter) matches the Reynolds number of the animal tests (Re ≈1200). This set of physical conditions is our "standard plume". The salt solution itself starts at 1 M and is diluted with 23 % Ethanol resulting in a 0.76 M NaCl source solution which is neutrally buoyant in fresh water. Crystal violet dye is added to make the plume visible on overhead video cameras.

Once a plume is established, Robolobster is placed inside the plume a meter downstream from the source jet. With the jet continuously injecting stimulus solution to maintain the plume in steady-state, the robot begins its orientation run. The trials reported here always started from this same point. The runs described in the next section lasted for 60 seconds each. In the sensor separation trials the run duration was 20 sec. During this time the conductivity inputs to each sensor and the gyroscope output are continuously logged.

3.2 Two Algorithms

Using two simple algorithms we have begun our exploration of both the plume and the capacity of dual inputs to guide Robolobster to the salt source. These algorithms are not intended to be lobster-like. We have intentionally designed them to be simple so that we can make progress towards understanding the interaction of the robot and the statistical rules of the turbulent plume. We want to exclude simple algorithms first and then increasingly complex ones. Ideally, this process of exclusions will leave us with a small set of competent algorithms of the minimum required complexity, and an understanding of the reasons for failure and success.

The behavioral repertoire of Robolobster consists of four behaviors: moving straight forward, straight backward, turning left and right. Forward motion is effected by supplying power to the wheels sufficient to move the robot in a straight line at a speed comparable to that of the lobster (9cm/sec). Backward is the same except that the direction of wheel rotation is reversed. Turns are

effected by stopping the wheel on the side of the desired turn. With these behaviors we specified two algorithms:

Algorithm 1 is specified by two rules:

1) If the difference between the left and right sensor signals exceeds a conductivity value of 9 μS the robot turns toward the side of the sensor receiving the higher concentration signal.

otherwise:

2) The robot moves forward with constant speed.

Algorithm 2 is the same as 1 with the addition of a third rule.

3) If the sum of the two sensor inputs drops below 7 μS, the two motors reverse causing the robot to back up at half speed.

The addition of the back-up rule in algorithm 2 keeps the robot inside the plume.

We evaluate the robot's performance using a number of measures. Only two will concern us here: 1) The overall proportion of successful trials is scored: and 2) the time to the first successful hit is scored. A successful trial (or hit) occurs when the robot passes within 5 cm of the source.

3.3 Robot Starting Orientation in Turbulent Salt Plumes

In our early experiments we placed the robot 90 cm from the source and pointing directly toward the source. In this starting condition, Running Algorithm 1, Robolobster found the source about a third of the time (n=10) with Algorithm 1 and about two thirds of the time (n=10) with Algorithm 2. Given what we know about the patchy nature of our "standard plume" it is not surprising that the "hit rate" was less than perfect: a smooth concentration gradient does not exist in the standard plume (FIGURE 2).

Changing the starting orientation of the robot markedly decreased the number of hits obtained by both algorithms. When the robot began a trial pointed obliquely to the source (45 degrees from the long axis of the flume) Robolobster with Algorithm 1 did not find the source (n=10). With Algorithm 2 it only found the source on one trial in ten (FIGURE 3).

The poor overall performance of these algorithms is not surprising given the character of the plume (FIGURE 3). It was clear to observers that during successful trials (trials with hits) there was a marked change in the behavior of the robot as it approached the source. Far downstream, both algorithms led the robot on paths which showed numerous turns at irregular intervals, including backward motions for Algorithm 2. Closer to the source both algorithms showed a series of broad and smooth counter-turns that zeroed in on the source. Here, where plume exits were obvious, the robot was tracing the border of the proximal region of the salt jet. Once inside this proximal jet, Algorithm 2 found the source in three quarters of the trials. In the remainder, the 60 sec run duration ended before the source was reached. Algorithm 1 scored hits in about half the trials. In the proximal jet the robot generally failed to score a hit with Algorithm 1 because it exited the plume and continued moving away thereafter. The robot running Algorithm 2 never lost the plume and in general failed simply because it ran out of run time. On all occasions it appeared that given sufficient time the robot running Algorithm 2 would eventually have found the source.

Algorithm 1 loses the plume because it lacks the means to return when both sensors fall out of the plume. This is most obvious in the region of the proximal jet where the plume is narrowest. Of course the back-up condition in Algorithm 2 corrects for this problem providing for greater accuracy, allowing the robot to trace the boundary of the plume in the proximal jet (FIGURE 4a and b).

The increased accuracy of Algorithm 2 is bought at a price of diminished speed (FIGURE 4c and d). This leads to an interesting parallel between the behavior of the robot and the lobster. As a real lobster approaches the source, it first searches about slowly and in many directions. Then it picks up speed and locates the source in a series of gentle path curves, occasionally in an almost straight path (Moore *et.al*,1991; Basil *et.al*,1996 in prep). The transition in robot behavior from tentative jerking paths in the distal patch field to smooth graceful turns in the proximal jet mirrors this behavior. We suggest that the speed and orientation changes reflect a change in the complexity of the problem and requires a shift in strategy for both lobster and robot. We will return to this question in the discussion.

3.4 Sensor Separation

One expects the separation between the sensors to have a dramatic impact on the competence of the orientation algorithm. Sensors placed too closely would provide no more information than a single sensor; wider separation might provide a more accurate estimate of a smooth gradient. However, in the distal patch field where smooth gradients do not exist, greater separation of the two sensors does not solve the crucial problem of reducing the time required to arrive at average concentration estimates. We therefore analyzed the temporal information the robot obtained from pairs of sensors placed at different separations.

We conducted Algorithm 2 trials with the conductivity sensors separated by 1,3 or 9 cm: compared to the 6 cm separation used in the orientation experiments

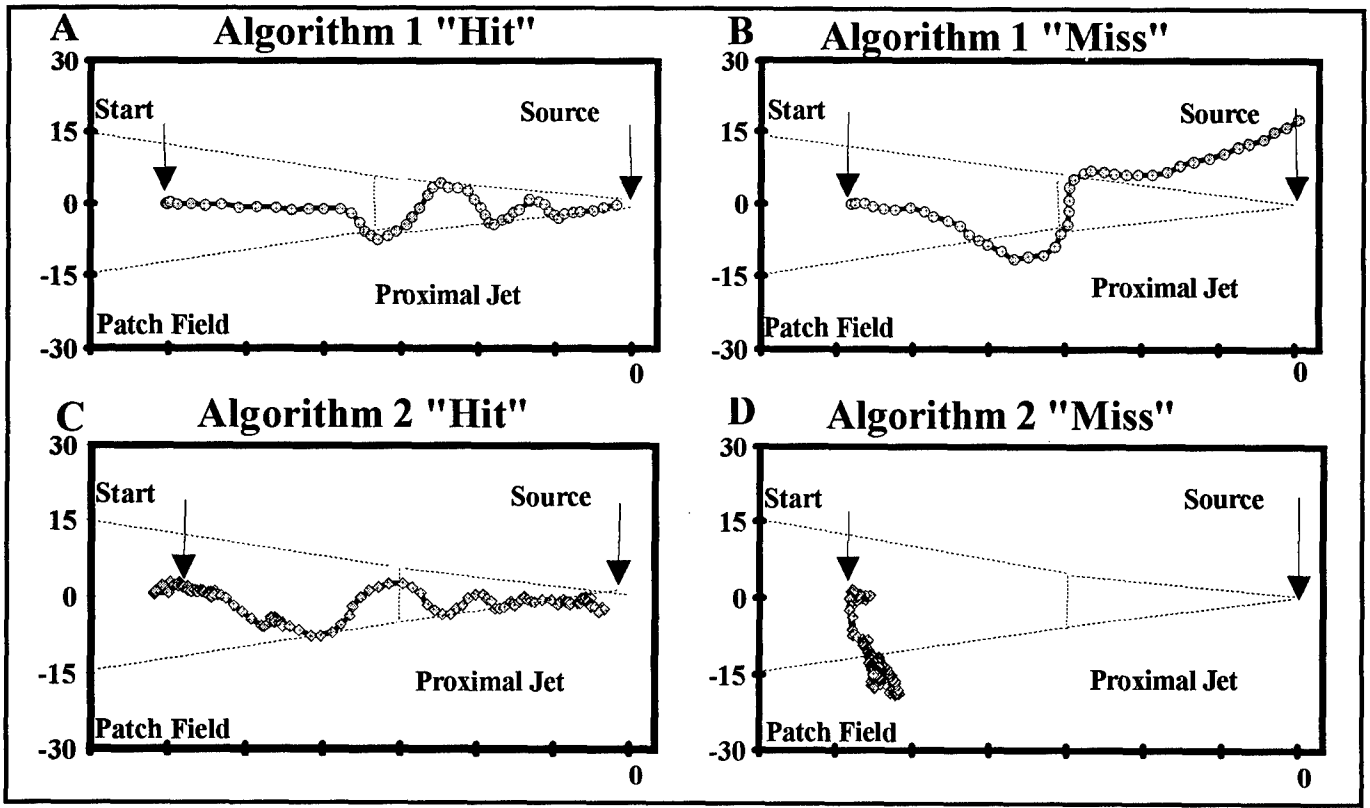


FIGURE 3. Successful (**a**) and unsuccessful (**b**) paths produced by Robolobster running Algorithm 1. Note in (b) that the robot exits the plume: lacking additional input it proceeds in a straight line. Successful (**c**) and unsuccessful (**d**) paths from Robolobster running Algorithm 2. Note in (c) that the robot began the run by backing away from the starting position. In (d) the robot spends the majority of its 60 second run moving back-and-forth at the right plume boundary in the patch field. Each position is marked after 1/3 second has elapsed. Note that in (d) the robot runs out of time before it reaches the proximal jet. Distances are shown in centimeters (15 cm ticks) zeroed at the source on each axis. The approximate outline of the plume is indicated with dashed lines. A dashed line also separates the approximate boundary of the proximal jet and the patch field.

described above. The results hint at a solution to the problem of navigation in the distal patch field.

FIGURE 6 shows the average cross correlation function (CCF) of the left and right sensors while the robot was negotiating the distal patch field. The traces plot the average CCF for each of the three sensor spacings (1, 3, and 9 cm). The delay of the CCF maximum for each separation is located at zero because the control algorithm (1 or 2) steers the robot in a way that maintains the minimum difference between the sensor inputs. The interesting result is that the magnitude of the zero delay cross correlation falls off with increasing separation. The correlation is strongest when there is little separation between the sensors (i.e., virtual autocorrelation). We believe that increased sensor separation decreases the cross correlation because as the separation increases the peak shape similarity between the encountered salt patches decreases (i.e., the simultaneous concentration signals are less similar at greater distances).

At 9 cm separation the sensors are on average weakly correlated in the distal patch field. The sensor information is largely independent of the local patch/pulse structure. At the narrowest separation (1 cm) the two input signals are virtually identical as indicated by the harmonics in the CCF (the periodicity of 3 sec probably results from the back-up and turning cycles of the robot). At 3 cm, however, the cross correlation is strong and the harmonics are not evident. The lack of harmonics indicates that on average the two sensors capture different information about the local patch structure; strong zero delay cross correlation indicates that at this separation the scale of the local patch structure has not been exceeded. What makes this result intriguing is the coincidence that a 3 cm separation corresponds to the antennular separation of lobsters.

4. Discussion

Modeling odor dispersal in turbulence is a complex and computationally intense undertaking and no generally accepted models exist. Consequently, investigations of animal chemo-orientation have typically been computer simulations in which both the organism and the environment were simulated. Typically these studies assume plumes with smooth gradients (Giles et.al. 1995, Pentcheff et. al. 1994, Beer and Chiel,1991) or idealized patches (Belanger, 1996) without reference to the actual

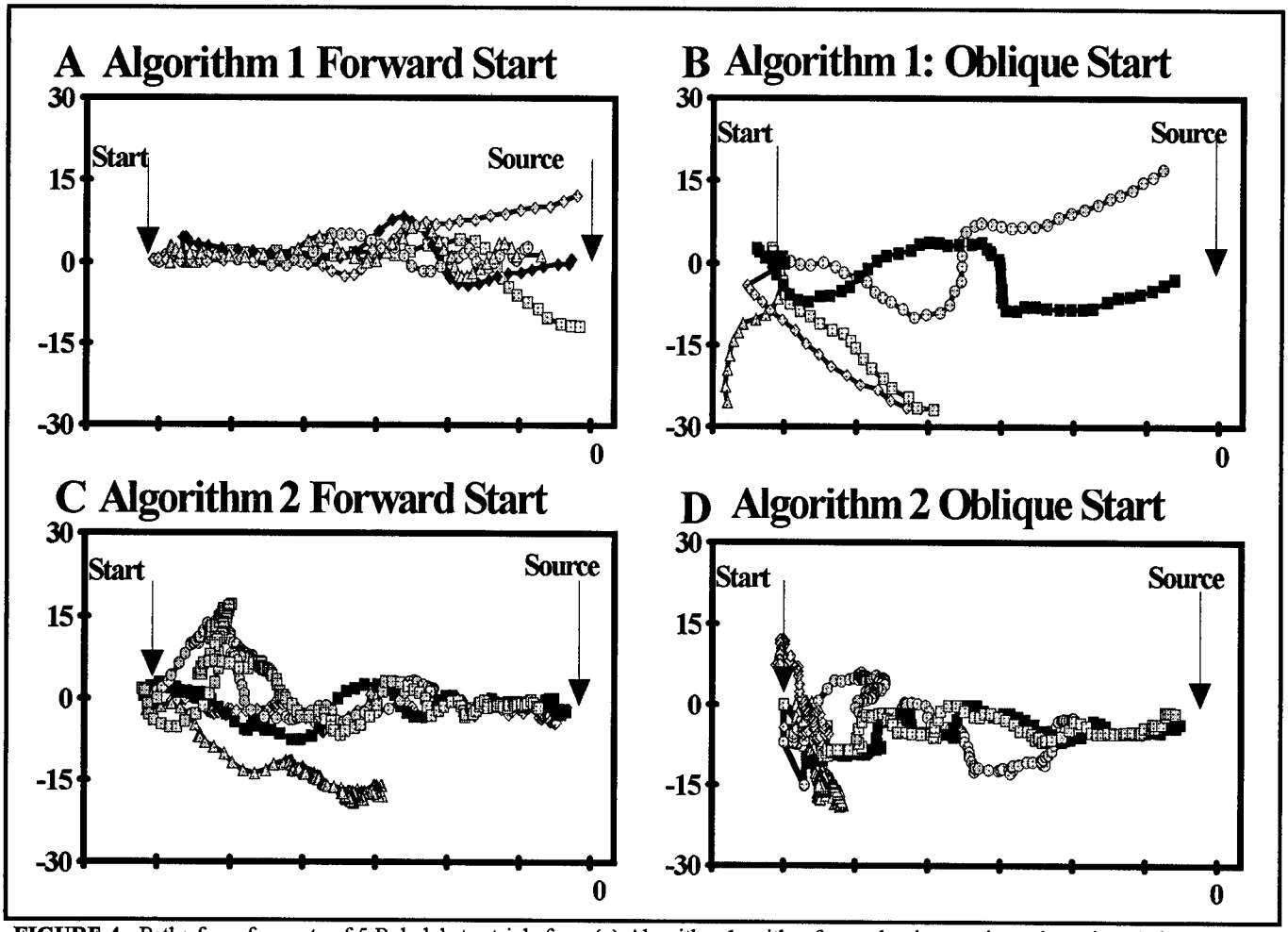


FIGURE 4. Paths from four sets of 5 Robolobster trials from (**a**) Algorithm 1, with a forward point starting orientation (0 degrees from the flume long axis directly downstream from the source.) (**b**) Algorithm 1, with an oblique starting orientation (45 degrees right from the flume long axis and directly downstream from the source), (**c**) Algorithm 2 with a forward starting orientation and (**d**) Algorithm 2 with and oblique starting orientation. Note that Algorithm 1 sometimes exits the plume from within the proximal jet while Algorithm 2 always remains within the proximal jet (though run duration may be exceed there before it reaches the source). Note also that with an oblique start Algorithm 2 can expend the entire mission duration wandering in the patch field. Algorithm 1 paths have been truncated to exclude those points where the robot proceeded upstream beyond the source. In these cases the robot continued on a straight line until the run duration was reached. The axes are scaled as in FIGURE 3.

statistics of odor dispersal. We have chosen the alternative of studying chemo-orientation strategies in which the animal but not the odor distribution is simulated.

Our investigations of simple orientation algorithms in turbulent salt plumes have identified two distinct regions of the plume: a proximal jet where the orientation task is easy and a distal patch field where it is more difficult. The difficulty of negotiating the distal patch field on purely chemical cues results from the discontinuous distribution of the salt patches. The patches themselves do not provide a smooth gradient pointing to the source and (from the lobsters point of view) waiting to average the approaching patches to estimate the gradient takes too long to be practical.

There were two general reasons Algorithm 2 failed to guide the robot to the source, 1) the mission duration ended with the robot frittering away its time near its starting point in the distal patch field or 2) the robot followed a single patch of odor in a jerky path downstream *away* from the source. Both of these results indicate a lack of timely, directional information available to the robot in the distal patch field. Yet, blindfolded, orienting lobsters behave as if they "know" the direction of the source even when they are in the distal patch field (Basil and Atema 1994, Basil *et.al*,1996 in prep).

Within the proximal jet the patches are dense and frequent enough to allow an easy and timely indication of whether a sensor is inside or outside the plume. Here the simple back-up instruction of Algorithm 2 is sufficient to keep the robot inside the plume. The narrowing of the proximal end of the jet and the increasing density of patches eventually draws the robot to the source. Further

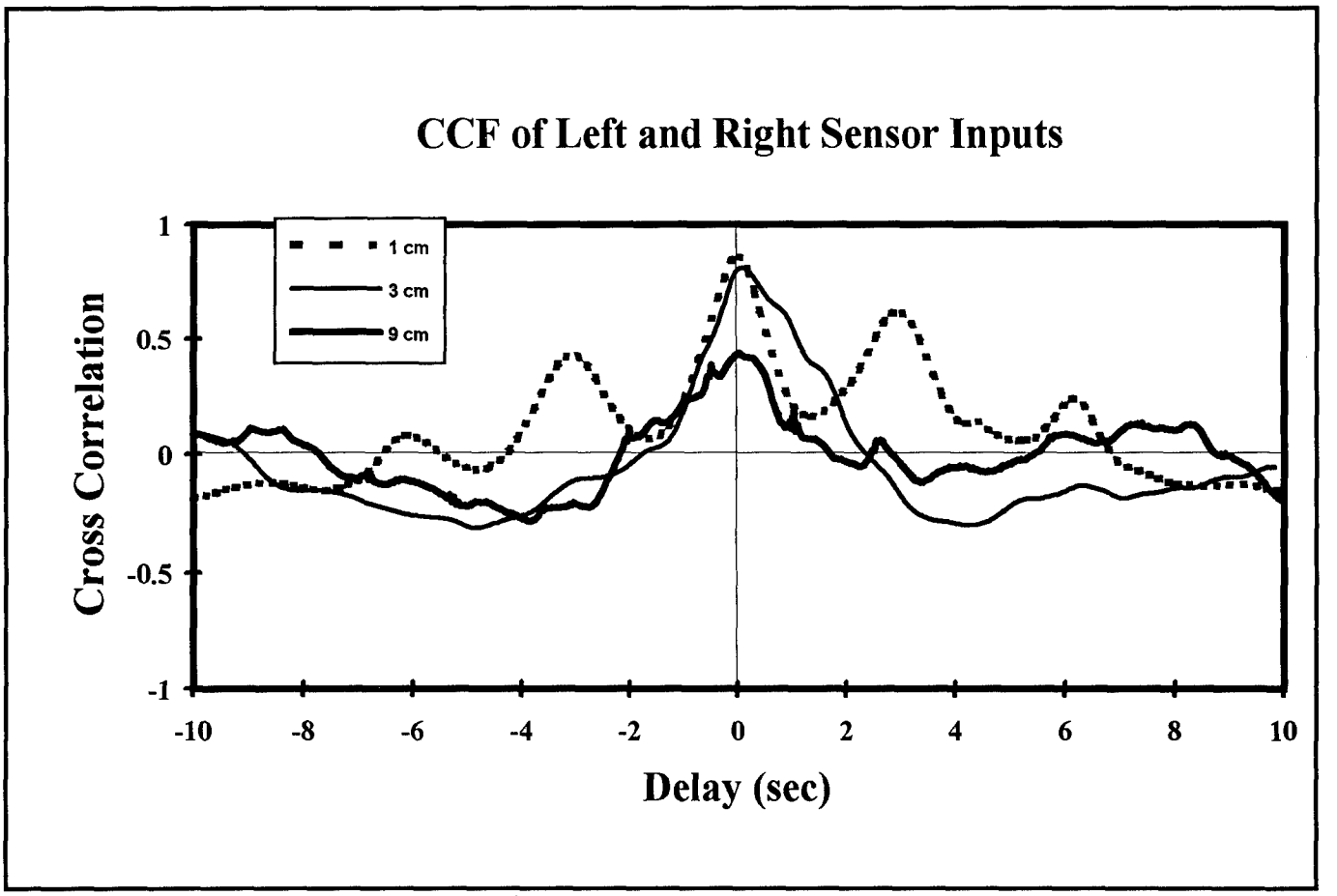


FIGURE 5. Cross correlations of the left and right input channels at different sensor separations (1, 3 and 9 cm). Maxima of all three CCFs occur at or near zero delay. Note the harmonics in the 1 cm CCF (dashed line) and their absence from the 3 and 9 cm CCFs.

downstream the patches are more sparse and Robolobster shows irregular, wandering paths (FIGURE 4B). Thus, it seems that in the distal patch field the algorithm guiding the robot needs a source of directional information greater than the instantaneous left-right difference. Our work with the robot and the physics of odor plumes gives us four avenues of further research to pursue the goal of matching lobster competence at locating turbulent odor sources:

Rheotaxis: With the failure of Algorithm 2 to detect a sense of direction from patches might have been ameliorated if the robot had simply known to head upstream. Eventually, this would lead to the plume boundary and from there the robot could track the edge of the plume. This type of behavior is not observed in lobsters but is evident in moths (Murlis, J., J. Elkinton and R. Carde 1992; Elkinton J., R. Carde, and C. Mason, 1984) and blue crabs (Zimmer-Faust, Finelli, Pentcheff and Wethey, 1995). In general, lobsters' heading angles decrease as they approach the source (Moore, *et.al.*,1991) and rheotaxis (walking into the mean flow) implies that the heading angle increases as the source is approached. This does not mean that lobsters (or Robolobster) could not make use of this cue in conjunction with some other mechanism particularly considering local flow (*eddy rheotaxis*, Atema 1996).

Peak Parameter Feature Extraction: Studies of plume structure indicate that the parameters of the peaks (temporal profiles of concentration patches) themselves actually point to the source (Moore and Atema, 1991, Dittmer, Grasso and Atema 1995, Grasso, Basil and Atema 1996 in prep). Thus, for example a single sensor following a gradient of rising slopes of encountered odor patches would eventually encounter the source. There are a number of such candidate peak features which might be usable singly or in combination. For that reason this algorithm requires some form of memory of the progression of features encountered. Statistical analysis of plume patch parameters demonstrates that it is possible however to use the shape of the patches/pulses to determine location relative to the source.

Memory Mechanisms: By memory mechanisms we mean allowing the robot to track it's progress though the course of a trial. Some sensor derived performance measure might allow the robot (or a lobster) to avoid following a path that is obviously wrong. Memory mechanisms will improve the performance of any algorithm

by imposing a hierarchical structure on the navigation decisions: it could be as simple as returning to the start if the plume is lost or as complex as tracking of encountered patch shapes in comparison with a stored map of expectations.

Bilateral Comparison: The behavioral evidence indicates that the lobster uses both of its antennules to its advantage and suffers a severe performance deficit when deprived of the use of one of them. The question is how does it use them? The observation that the separation of antennules in our experimental animals falls into a range that preserves the local patch structure in our plume suggests that the animal might be using the shapes of the patches to determine the correct direction. One application of this would be to use a cross-correlation of the two signals to guide the animal.

5. Conclusion

The results of a successful solution to the distal patch field problem will be more than a better understanding of how lobsters locate food. Lobsters and most other macroscopic animals live in an environment in which turbulence is an omnipresent force which shapes the temporal pattern of arriving chemicals. An understanding of how lobsters orient under turbulent flow conditions is a step toward understanding chemotaxis mechanisms in biology and engineering. It affords the ability to locate point sources of pollution and chemical hazards under these conditions. It also represents a new and previously unexplored way of sensing and understanding the physical environment.

Acknowledgments

This work is supported by an NSF grant BES-9315791 to JA and a subcontract to C. Chryssostomidis at MIT Sea Grant. We thank Jaimie Cho, Jonathan Dale, Paul DiNunno, Kevin Dittmer and Diana Ma for technical assistance with the experiments and data analysis. And Dr. Jennifer A. Basil and Claire Balint for proof reading the manuscript.

References

Atema, J. (1996) "Eddy Chemotaxis and Odor Landscapes: Exploration of Nature with Animal Sensors." Bio. Bull. (in press).

Atema, J. (1995) "Chemical Signals in the Marine Environment: Dispersal, Detection and Temporal Signal Analysis." PNAS, 92(1):62-63.

Atema, J. (1985) "Chemoreception in the Sea: Adaptations of Chemoreceptors and Behavior to Aquatic Stimulus Conditions." Soc. Exp. Biol. Symp. 39:387-423.

Basil and Atema, (1994) "Lobster Orientation in Turbulent Odor Plumes: Simultaneous Measurement of Tracking Behavior and Temporal Odor Patterns." Bio. Bull. 187:272-273.

Basil, J., F. Grasso and J. Atema (1996) "High Resolution Bilateral Odor Tracer Measurement During Lobster (*Homarus americanus*) Orientation Behavior in a Turbulent Odor Plume." (in prep)

Beer, R., and H. Cheil (1991) "The Neural Basis of Behavioral Choice in an Artificial Insect." From Animals to Animats: Proceedings of the First International Conference on Simulation of Adaptive Behavior. MIT Press, Cambridge, MA. 247-254.

Belanger, J. (1996) "Behavioral Strategies Underlying Odor-Mediated Flight in Moths: Lessons from Simulations." East Coast Nerve Net (Talk). April 22-23 1996.

Bursell, E. (1987) "The Effect of Wind Borne Odors on the Direction of Flight in Tsetse Flies (*Glossinas spp.*)." Physiol. Entomol. 12:149-156.

Bursell, E. (1984) "Observations on the Orientation of Tsetse Flies (*Glossina pallidipes*) to Wind Borne Odors." Physiol. Entomol. 9:133-137.

Consi, T., J. Atema, C. Goudey, J. Cho and C. Chryssostomidis (1994) "AUV Guidance with Chemical Signals." Proceedings of the IEEE Symposium on Autonomous Underwater Vehicle Technology. Cambridge, MA July 19-20,1994.

Devine, D. and J. Atema (1982) "Function of Chemoreceptor Organs in Spatial Orientation of Lobsters (*Homarus americanus*): Differences and Overlap." Biol. Bull. 163:144-153.

Dittmer, K., F. Grasso, and J. Atema (1995) "Effects of Varying Plume Turbulence on Temporal Concentration Signals Available to Orienting Lobsters." Biol. Bull 189:232-233.

Elkinton J., R. Carde, and C. Mason (1984) "Evaluation of time-average dispersion models for estimating pheromone concentration in deciduous forest." J. Chem. Ecol. 10: 1081-1108.

Giles, E., C. McInnes and D. Neil "Emergent Search Behavior in a Sensory Gradient modeled by a Simple Artificial Animal." IN: Nervous Systems and Behavior: Proceedings of the 4th international Congress of

Neuroethology. Sept. 3-8,1995 Cambridge U.K. Malcom Burros and Thomas Matheson Eds.

Gomez, G., R. Voigt and J. Atema (1994) "Frequency Filter Properties of Lobster Chemoreceptor Cells: Effects of Stimulus Concentration on the Coding of Pulse Trains." J. Comp. Physiol. 174:803-811.

Grasso, F., J. Basil and J. Atema (1996) "Directional Information in the Dynamic Structure of a Turbulent Odor Plume Measured with A Pair of Lobster-Scaled 'Odor' Sensors: Potential for Eddy Chemotaxis." (in prep)

Mafra-Neto, A., and R.T. Carde (1994) "Fine-scale structure of pheromone plumes modulates upwind orientation of flying moths." Nature 369: 142-144.

Moore, P., Scholz and J. Atema (1991) "Chemical Orientation of Lobsters *Homarus americanus* in Turbulent Odor Plumes." J. Chem. Ecol. 17(2): 1293-1307.

Moore, P., and J. Atema (1991) "Spatial Information Contained in Three-Dimensional Fine Structure of an Aquatic Odor Plume." Bio. Bull. 181:408-418.

Murlis, J., J. Elkinton and R. Carde (1992) "Odor Plumes and How Insects Use Them." Ann. Rev. Entomol. 37:505-532.

Pentcheff, N., C. Finelli, D. Wethey and R. Zimmer-Faust (1993) "Turbulent Odor Plumes and Chemo-Orientation in Nature." Sixteenth Annual Meeting of the Association for Chemoreception Sciences. #105. Sarasota, Florida April 13-17,1996.

Vickers, N.J., and T.C. Baker (1992) "Male *Helothis virescens* maintain upwind flight in response to experimentally pulsed filaments of their sex pheromone (Lepidoptera: Nocturidae)". J. Insect Behav. 5: 669-687.

Zimmer-Faust, R., C. Finelli, N. Pentcheff and D. Wethy (1995) "Odor Plumes and Animal Navigation in Turbulent Water Flow: A Field Study." Bio. Bull. 188:111-116.

Dynamics for vision-based autonomous mobile robots

Hartmut Neven, Axel Steinhage, Martin Giese, Carsten Bruckhoff
Institut für Neuroinformatik, Ruhr-Universität Bochum
44780 Bochum, Germany
Tel: +49 234 7007975
Email: neven/axel/carsten/giese@neuroinformatik.ruhr-uni-bochum.de

Abstract

We present a dynamical systems approach to generate basic navigation behaviors implemented on an autonomous mobile vehicle. In this approach, a behavior is expressed as a stable state of a variable in a dynamical systems equation. The value of this behavioral variable is specified by incoming information which is provided by the sensory systems typically but can also be determined by other behavioral levels. Multiple sources of information act as forces in the vector field of the dynamical equation that describes a specific behavioral level. Each force votes for a certain *value* of the behavioral variable, it has a *strength* that represents the relative importance or certainty of the specified information and it is supplied with a *range*, that determines the overlap between the multiple incouplings. As an example for this approach on the sensor side, we describe a dynamical optical flow estimation that takes advantage of the favorable stability properties of this type of dynamical systems as robust estimators particularly suited for ambiguous and noisy visual information. On the effector side we present an obstacle avoidance scheme that employs a dynamical system for steering a visually guided mobile robot platform in an indoor environment with obstacles. By coupling of the sensor- and the effector system, a closed loop navigation behavior is achieved.

1 Introduction

1.1 The dynamic approach

The dynamic approach to behavior generation is based on the idea, that behaviors can be expressed as stable states of dynamical systems (see Schöner and Dose 1992, Schöner et al. 1995 for a review). The variables that determine the actual state of a robot, like forward velocity or heading direction for instance, are treated as behavioral variables. Desired behaviors, like approaching a target in space, come about by designing the dynamical system such that the corresponding behavioral variable has a value that belongs to the set of stable solutions of the dynamical systems equation. In the case of target acquisition for instance, the dynamical system is designed such, that the heading direction of the agent that points to the target, is a stable fixed point or *attractor* of the system's differential equation. Behaviors, that are not desired, are avoided, by designing the dynamical system such, that the corresponding values of the behavioral variables belong to the set of unstable fixed points of the systems equation. For the behavior of obstacle avoidance for instance, the heading directions, that point to the obstacles are made unstable fixed points or *repellors* of the equation.

While generating desired behaviors, the behavioral variables that describe the agent's state have always values, that are stable solutions of the dynamical systems equation. Here, a problem seems to come out: how can a sensible behavior arise, if the control variable is in an attractor at all times ? This problem is solved by designing the time scales of the system such that its attractors move but on a slower time scale than the behavioral variable itself. Under these circumstances, the qualitative theory of dynamical systems, such as analyzing bifurcations and stability can be used to design the behavioral dynamics (Schöner et al. 1995). This approach of letting the system be in an attractor at all times is an essential difference compared to the potential field method (Arkin 1990) that works primarly with transients and defines an attractor for the terminal state only. This implies, that here control and decision making are not endowed with stability properties.

As in the example of obstacle avoidance, there may be situations, in which the dynamical system has to deal with more than one contribution. Depending on the relative importance, every contribution is coupled into the dynamical system with a specific *strength*. An obstacle that is far away contributes to the dynamical system of heading direction with a lower strength than the one of an obstacle that is near.

Not only the strength of multiple contributions is important but also their *range* of influence. Let us imagine

a situation in which two identical obstacles are right in front of the robot with the same distance to the robots sensors. Between the obstacles there is a narrow gap, that is too small for the robot to go through. Naturally, the obstacle on the left would vote for a turn of the robot to the right and vice versa. Simply summing the contributions would always lead to an attractor between the two obstacles no matter how narrow the gap is. In absence of other influences the robot would crash into the obstacles. This problem can be solved in an elegant way by defining an overlap between the contributions: every single contribution is provided with a *range* of influence. In the differential equation this can be done by multiplying every contribution with a gaussian profile for instance. This leaves the stability properties of the attractor unchanged. The width of this function defines a critical distance between the contributions: for distances below this width contributions are averaged and thus treated as one. Distances higher than the critical one lead to a decision of the system: if the current state of the behavioral variable is far away from one of the two values defined by the contributions, this contribution has no influence on the behavior.

An example for a dynamical system that fits in this scheme is the following simple system for controling the heading direction x of a robot in an environment with N targets:

$$\dot{x} = \sum_{i=1}^{N} \lambda_i (x_i - x) \exp(-\frac{(x_i - x)^2}{2\sigma_i^2}) \qquad (1)$$

Here λ_i and σ_i define the relative strengths and ranges of influence for each target contribution, while the x_i are the heading directions in which the targets can be seen from the robots location. Fig. 1 shows an example for two contributions, x_1 and x_2

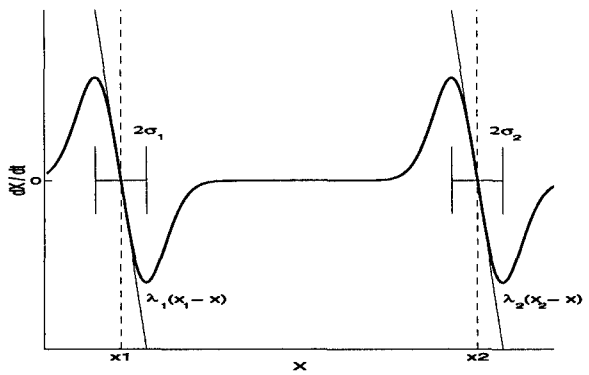


Figure 1: Phaseplot of the simple dynamical system in (1) with two attractive contributions

The λ_i and x_i can be specified by sensor input. For continuous input functions $s(x,t)$ and fixed range of influence c we can write the dynamics in the following way:

$$\dot{x} = \frac{1}{\tau_x} \int_{-\infty}^{\infty} s(x',t)(x' - x) \exp(-\frac{(x' - x)^2}{c_x^2}) dx' \qquad (2)$$

Here, τ_x is the time-scale of the dynamics. For discrete sensor input the stimulus $s(x,t)$ can be written in the form

$$s(x,t) = \sum_{i=1}^{n} \lambda_i \delta(x_i - x) \qquad (3)$$

with the *Dirac* $\delta()$-distribution. This leads to an equation of the form (1).

The ideas presented so far have been successfully demonstrated in several theoretical and practical robotics applications (Schöner and Dose 1992, Neven and Schöner 1995, Steinhage and Schöner 1995, Bicho and Schöner 1996) and have been extended to a neural field dynamics in (Schöner and Engels 1994).

1.2 *The dynamical approach from the viewpoint of decision theory*

Before we show two typical examples of such applications let us discuss a new aspect of this theory: we want to look at equation (2) from the point of view of decision theory. The formulation of decision theory is to introduce a space of actions or alternatives A, a set containing the possible states of nature S and a utility function $U(\varsigma, a)$ which assigns a utility for taking action a when the state of nature is ς. Of course ς is only accessible via measurements $m \in M$ taken by elementary detectors. A decision policy

$$\mathcal{D} : M \to A \qquad m \mapsto a \qquad (4)$$

decides for an action a given the measurement vector m. We will discriminate between two cases: decision making on the perceptual level and decision making on the behavioral level. On the perceptual level a comparison with the Bayes estimator will be given, while on the behavioral level the dynamic approach is discussed as an arbitration scheme.

1.2.1 *Decision making on the perceptual level*

Estimation deals with the problem, that the state ς of a system in a natural environment is not directly knowledgeable to the system itself. Therefore an estimate of this state $\hat{\varsigma}$ has to be infered from a number of measurements m_i. An estimation technique, that is optimal under very general conditions and which allows an integration of a priori information about the state of nature is Bayesian estimation (Kay 1993). Let us assume we have a number of measurements $m_1, \ldots, m_n$ and a model $p(m_1, \ldots, m_n|\varsigma)$ that gives the conditonal probability that $m_1, \ldots, m_n$ will be measured given the system is in state ς. Further we have *a priori* knowledge about

the state ς which in the Bayesian language is given as an *a priori* probability distribution $p(\varsigma)$. Then a Bayesian estimate $\hat{\varsigma} \in S$ frequently employed is defined by the condition that it maximizes the *a posteriori* probability distribution

$$\hat{\varsigma} := \max_{\varsigma} p(\varsigma|m_1,\ldots,m_n) = \max_{\varsigma} \frac{p(m_1,\ldots,m_n|\varsigma)p(\varsigma)}{\sum_{\varsigma} p(m_1,\ldots,m_n|\varsigma)p(\varsigma)} \tag{5}$$

We restrict the discussion to the situation in which the individual detector measurements m_i are independent such that the measurement model modularizes into

$$p(m_1,\ldots,m_n|\varsigma) = p_1(m_1|\varsigma)\ldots p_n(m_n|\varsigma) \tag{6}$$

To obtain analytically evaluable expressions we assume further that the individual detector models as well as the prior information are modelled as gaussian probability distributions

$$p(m_i|\varsigma) = \frac{1}{N_i}\exp(-\frac{(m_i-\varsigma)^2}{2\sigma_i^2}) \tag{7}$$

$$p(\varsigma) = \frac{1}{N_0}\exp(-\frac{(\varsigma_0-\varsigma)^2}{2\sigma_0^2}) \tag{8}$$

In this case the maximum a posteriori estimator consits of a weighted average of the individual most probable estimates

$$\hat{\varsigma} = \frac{\frac{\varsigma_0}{\sigma_i^2} + \sum_{i=1}^{n}\frac{m_i}{\sigma_i^2}}{\sum_{i=0}^{n}\sigma_i^2} \tag{9}$$

The problem with this estimator is, that it is not robust because each individual measurement is taken serious and contributes to the final estimate. When having to deal with noisy environmental information provided by a visual sensor, this turns out to be a major drawback because outliers are quite frequent. A related problem is that finding the correct probability distributions is often exceedingly difficult because the space of states of nature can be very large for natural environments. Moreover, in a dynamic environment the probabilities can vary over time and it is often not possible to track this variation.

How do we deal with this problem in the dynamic approach? Here *a priori* information can be accounted for by setting a stimulus of the form

$$s_{pred}(\varsigma,t) = \eta_{\text{pred}}\delta(\varsigma-\varsigma_{pred}) \tag{10}$$

The actual state is assumed to be in the neighbourhood of the predicted state. The information provided by the n elementary detectors is expressed by stimulus contributions of the form

$$s_{\text{sensor}} = \sum_{i=1}^{n}\eta_i\delta(\varsigma-m_i) \tag{11}$$

Feeding the sum $s = s_{\text{pred}} + s_{\text{sensor}}$ into the dynamical equation (2) and resolving it for the fixed point $\dot{\varsigma} = 0$ gives

$$\hat{\varsigma} = \frac{\eta_{\text{pred}}s_{\text{pred}}e^{-\frac{(\varsigma_{\text{pred}}-\hat{\varsigma})^2}{c_\varsigma^2}} + \sum_{i=1}^{n}\eta_i m_i e^{-\frac{(m_i-\hat{\varsigma})^2}{c_\varsigma^2}}}{\eta_{\text{pred}}e^{-\frac{(\varsigma_{\text{pred}}-\hat{\varsigma})^2}{c_\varsigma^2}} + \sum_{i=1}^{n}\eta_i e^{-\frac{(m_i-\hat{\varsigma})^2}{c_\varsigma^2}}} \tag{12}$$

This is an implicit equation to determine $\hat{\varsigma}$. The actual state will decide, which one is chosen. This estimate is robust in the sense, that state measurements which are far away from the stable estimate are weighted such, that they do not contribute to the estimate and as a consequence are discarded as outliers.

Thus neural fields can be regarded as a particular instance of a robust estimator that bases the rejection of outliers on evidence accumulated over time. (A classical reference on the topic of outliers is Barnett and Lewis 1987).

We close this section by summarizing conditions under which the use of a neural field estimator should be considered.

i The estimation of a current state of nature is based on a series of sweeps of measurements. A single sweep consists of a number of sensor readings, each of which conveys information regarding the state of nature.

ii An initialization of the neural field has to be possible. It is advantageous if once in a while information is unambigous, i.e. only a single attractor exists. These situations lead to a recalibration of the estimator.

iii The detector answers come along with confidence measures that can serve the determination of the η_i.

1.2.2 Decision making on the behavioral level

One of the problems of decision making on the behavioral level is, that the space of states of nature S to be considered in order to take action a is in general far too large to construct explicitly the utility function $U(\varsigma, a)$. In addition to the large dimensionality this function will be in general nonlinear. Because of these difficulties no general design methodology exists and consequently many approaches evolved. Probably all of them share the first step to modularize the set of states of nature

$$S = S_1 \otimes S_2 \otimes S_3 \ldots \otimes S_N \tag{13}$$

and to concentrate on the construction of the partial utility functions

$$U_i : S_i \to A \qquad \varsigma_i \mapsto a \tag{14}$$

Splitting up the state space entrains the problem of how to combine the outcomes of the partial decision rules $\mathcal{D}_i()$. Thus, a need for arbitration schemes arises.

Constructing a single decision function $\mathcal{D}_i()$ is often equivalent to solving an ordinary control problem. Difficulties arise, when at least two decision rules vote for conflicting actions. Here the concept of priority is often introduced. The priorities assigned to the different alternatives are often functions of sensor measurements. If high noise levels contaminate the measurements m_i, the calculated priorities tend to fluctuate strongly leading to numerous switches between alternatives and no coherent behavior results. This problem is acutely felt in the robotics community (Holland 1995). A number of arbitration schemes have been proposed.

The arbitration between different votes using the dynamic approach proceeds as follows: If two alternatives a_1 and a_2 are given with priorities η_1 and η_2, then the stimulus

$$s(a) = \eta_1 \delta(a - a_1) + \eta_2 \delta(a - a_2) \tag{15}$$

governs the decision. This is exactly the situation discussed before: The initial state determines which value of a applies dependent on the range c_a. However, this bistable dynamical system displays hysteresis if the priorities η_i change. In the context of decision making one can say, that hysteresis stabilizes a decision. This is one virtue of decision making using a dynamical system. Another problem for many other arbitration schemes is, that the accuracy of reaching one goal is often degraded by attempting to fullfill also another goal. The underlying question is: when is it sensible to compromise ? The range limiting functions in the dynamical approach insure that votes for "far away" values do not affect the accuracy with which the goal is pursued.

In the following parts of the paper we present two examples of our approach in detail: on the perceptual side we describe a method for dynamical estimation of looming flow and on the effector side we present an approach to obstacle avoidance behavior. They are both parts of a navigation system that has been implemented on the autonomous robot vehicle MARVIN at the *Institut für Neuroinformatik, Ruhr-University of Bochum, Germany.*

2 Dynamical estimation of looming flow

2.1 Correlation based vision

A direct way to infer spatial structure from time-varying images is to compute optical flow. As a point of reference we depart from the framework povided by the correlational algorithm proposed by Little et al. 1988 (see also Bülthoff et al. 1989 and Bohrer et al. 1990) to compute optical flow since this method is computationally the leanest of the known approaches (Barron et al 1994).

In the following $p = (h, v)$ signifies a point in an image and $I(p)$ its greyvalue. To determine the correspondence vector at a given point $p_0 = (h_0, v_0)$ in a first image $I_1(p)$ the algorithm defines a quadratic test region $P_1(p)$ around this point.

$$P_1(p) = W(p - p_0) I_1(p) \tag{16}$$

where $W()$ is a windowing function given by

$$W(p) = \begin{cases} 1 & : \ |h| < C \text{ and } |v| < C \\ 0 & : \ \text{else.} \end{cases} \tag{17}$$

The algorithm determines the shift $dp = (dh, dv)$ which maximizes the correlation between the test region and an image patch $P_2(p, dp)$ in the second image $I_2(p)$

$$P_2(p, dp) = W(p - p_0) I_2(p + dp) \tag{18}$$

Correlation refers to the measure

$$\text{corr}(P_1(p), P_2(p, dp)) = N\left(\int |P_1(p') - P_2(p', dp)| dp'\right) \tag{19}$$

where the function $N()$ normalizes the negative of the sum of absolute greyvalue differences to the interval $[0, 1]$, so that 1 represents maximal confidence. Typically, not all possible shifts, but only those within a predefined search window are considered during maximization.

2.2 Comparing images such that flow vectors continuously varying in time have to be estimated

If one uses many consecutive frames to estimate optical flow in order to exploit the information contained in the time course, one has to formulate the problem of flow estimation such that temporal filtering is facilitated. To this end one has to take care that the parameter to be estimated varies continuously. Given continuous camera motion this can be accomplished by tracking the projection of a physical structure defined in a frame acquired at time t over several consecutive frames acquired at times $t' = t + \theta$ with $0 < \theta < \theta_{max}$. The vector $dp(p, t, t + \theta)$ which denotes the image motion at pixel p between time t and t' varies continuously in θ. This procedure is reinitiated for each time t. Of course in this way one obtains only average image velocities. But this is permissible if within the time interval $[0, \theta_{max}]$ the robot velocity remains approximately constant which can always be achieved by an appropriate choice of aquisition parameters.

A convenient way of computing the $dp(p, t, t + \theta)$ is to exploit the information contained in the time-varying correlation surface $\text{corr}(P_t(p), P_{t+\theta}(p, dp))$ by tracking the maximum which is located at $dp = 0$ for $\theta = 0$ while presenting several subsequent frames. In this case the test windows are only defined for time t and then matched with frames taken at times $t + \theta$. An alternative way is to measure correlations between any two

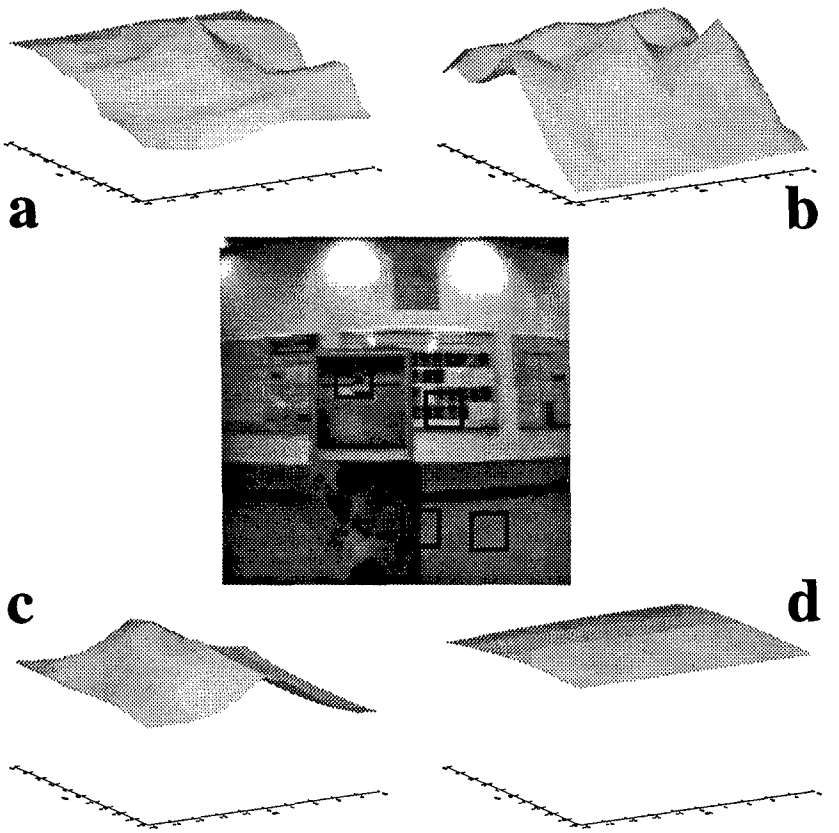


Figure 2: Correlation surfaces that result from comparing an image with itself. The squares denote the search windows over which correlations are determined.

successive frames. In this case the test windows are redefined for each time step and the correlation surfaces $\mathrm{corr}(P_{t+\theta}(p), P_{t+\theta+1}(p, dp))$ are exploited to calculate the $dp(p, t+\theta, t+\theta+1)$. Then the vector $dp(p, t, t+\theta)$ can be obtained recursively through

$$dp(p, t, t+\theta) = dp(p, t, t+\theta-1) + dp(p + dp(p, t, t+\theta-1), t+\theta-1, t+\theta) \quad (20)$$

with the initialisation $dp(p, t, t) = 0$. This alternative is problematic if one does not attempt to compute subpixel image velocities. In this case small image motions between consecutive frames can go undetected due to image discretisation. Note that small image velocities are to be expected for the desired motion vision which operates at high frame rates.

2.3 *What flow information can maximally be obtained from local image comparisons ?*

Before introducing the improvement obtained by making flow estimation dynamic we want to review what can maximally be achieved by local image comparisons (Haralick and Shapiro 1992). In the case of optical flow computation this discussion can be confined to the discussion of the correlation surface $\mathrm{corr}_{p,t,t}(dp)$ which results from a comparison of a frame acquired at time t with itself in the neighborhood of zero shifts $dp = 0$. (We facilitated the notation for the correlation surface in an obvious way.)

To simplify the discussion we assume that $\mathrm{corr}_{p,t,t}(dp)$ can locally be described completely by its development to the second order.

$$\mathrm{corr}_{p,t,t}(dh, dv) = \mathrm{corr}_{p,t,t}(0) + \begin{pmatrix} \frac{\partial \mathrm{corr}_{p,t,t}(0)}{\partial dh} \\ \frac{\partial \mathrm{corr}_{p,t,t}(0)}{\partial dv} \end{pmatrix}^T \begin{pmatrix} dh \\ dv \end{pmatrix} + \frac{1}{2} \begin{pmatrix} dh \\ dv \end{pmatrix}^T \begin{pmatrix} \frac{\partial^2 \mathrm{corr}_{p,t,t}(0)}{\partial^2 dh} & \frac{\partial^2 \mathrm{corr}_{p,t,t}(0)}{\partial dh \partial dv} \\ \frac{\partial^2 \mathrm{corr}_{p,t,t}(0)}{\partial dv \partial dh} & \frac{\partial^2 \mathrm{corr}_{p,t,t}(0)}{\partial^2 dv} \end{pmatrix} \begin{pmatrix} dh \\ dv \end{pmatrix}$$

A maximum of the correlation function is located at $dp = 0$ since

$$\mathrm{corr}_{p,t,t}(0) = 1 \quad (21)$$

Accordingly its first derivatives vanish at $dp = 0$

$$\frac{\partial}{\partial(dh)} corr_{p,t,t}(0) = \frac{\partial}{\partial(dv)} corr_{p,t,t}(0) = 0 \quad (22)$$

Analyzing this maximum leads to the discrimination of

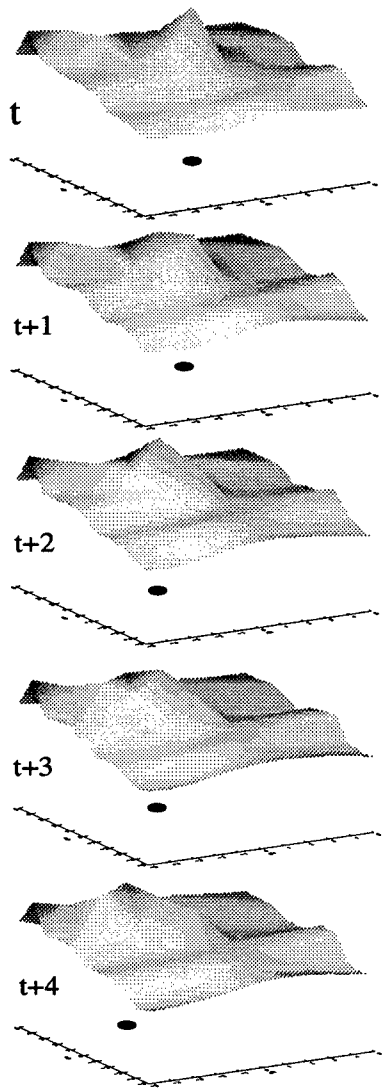


Figure 3: A dynamical system whose state (black dot) estimates the image motion (dh, dv). The three contributions s_{corr}, s_{aux} and $s_{\text{egomotion}}$ are added to form the overall stimulus. The field is spanned over the window denoted with a) in Fig. 2. Its size is 11×11 pixels. The origin of the field corresponding to a zero image shift is located at the center. Observe how the maximum located at $dp = 0$ for time $\theta = 0$ moves radially outwards due to a rectilinear robot motion into the direction of the optical axis. Since the state of the dynamical system "rides" on the local maximum that starts from the origin a spurious local maximum elsewhere will not affect its course.

three different cases according to whether the Hessian

$$H = \begin{pmatrix} \frac{\partial^2 \text{corr}_{p,t,t}(0)}{\partial^2 dh} & \frac{\partial^2 \text{corr}_{p,t,t}(0)}{\partial dh \partial dv} \\ \frac{\partial^2 \text{corr}_{p,t,t}(0)}{\partial dv \partial dh} & \frac{\partial^2 \text{corr}_{p,t,t}(0)}{\partial^2 dv} \end{pmatrix} \qquad (23)$$

has zero, one, or two vanishing eigenvalues. Fig. 2 illustrates these three situations for a typical image seen by the camera of our robot in its laboratory environment.

The first case in which the correlation surface shows a non-degenerate maximum can be observed for image regions with sufficient grey value variation such that the test patch is locally unique (see Fig. 2 a and b). The second situation is a result of greyvalue variations in only one direction. This situation leads to the well known aperture problem: a shift of the test patch can not be detected along the direction in which the greyvalue variation vanishes (see Fig. 2 c). Finally, the third situation comes about when locally around the pixel of interest no greyvalue variations occur at all. This results in a correlation function that is locally constant (see Fig. 2 d). Thus for affected pixels no flow information can be obtained from local image comparisons.

Properties of the correlation surface that are structurally stable remain under small variations of the camera position. It should be clear that optical flow is undefined along unstable directions in which the variation of the correlational surface vanishes since along these directions optical flow is extremely susceptible to noise. Adding a small noise function to the correlation surface can cause an abitrary shift of the flow vector into the structurally unstable directions. Thus it can now be stated in a well defined manner what optical flow information can be obtained by employing local operations only. *An optical flow field algorithm should yield the structurally stable components of the flow vector. The structurally unstable components should remain zero.* The last requirement renders the problem of optical flow computation well-posed. It is a heuristic that reflects the fact that in a navigation context even a small spurious flow vector component can have behavioral relevance. For instance when using looming flow as a cue for collision danger an arbitrarily small flow vector close to the focus of expansion can yield a small time-to-contact estimate that indicates immediate collision danger (for the definition of looming flow see Sec. 3). Thus erroneous flow vectors with a too large amplitude typically lead to the perception of "ghost obstacles".

To be more explicit a flow algorithm obeying the above requirements should retrieve the following information: In the first situation the full optical flow vector is computed. In the second situation the component into the direction of the eigenvector with the nonvanishing eigenvalue that is the so called "normal flow" can be determined while the component perpendicular to the normal flow remains zero. In the third situation the optical flow is undefined and accordingly the flow vector should stay zero.

2.4 Correlation surfaces as potentials for dynamical systems

Optical flow computation in the framework outlined above requires tracking the maximum of the time-varying

correlation surface $\mathrm{corr}(P_t(p), P_{t+\theta}(p, dp))$ which is located at $dp = 0$ for $\theta = 0$. To take advantage of the robust estimation properties of dynamical systems we define for each pixel a dynamical system over the space of possible shifts (dh, dv). The dynamical system employed has the form

$$\dot{dp} = \frac{1}{\tau_{dp}} \int_{-\infty}^{\infty} (dp' - dp) \exp\left(-\frac{(dp' - dp)^2}{c_{dp}^2}\right) \left(s_{\mathrm{corr}}(dp', \theta) + s_{\mathrm{aux}}(dp') + s_{\mathrm{egomotion}}(dp', \theta)\right) d(dp') \quad (24)$$

The initial condition is set to $dp = (0, 0)$. The relaxation time τ_{dp} has to be chosen such that

$$\dot{dp} >> v_{\mathrm{im}} \quad (25)$$

Hereby is v_{im} the maximal image velocity to be expected. The approach neccessitates that care is taken to keep v_{im} small, ideally smaller than 1 pixel/frame. Given a fixed acquisition rate which is hardware dependent this can be achieved through appropriate motion planning (see Section 3).

Three contributions couple into the dynamics. The contribution incorporating the correlation information is

$$s_{\mathrm{corr}}(dp, \theta) = \mathrm{corr}_{p,t,t+\theta}(dp) \quad (26)$$

The state of a dynamical system with the correlation term as the sole contribution tracks a nondegenerate maximum (see Fig. 3 and the Conclusions).

In the case that the maximum is degenerate the dynamics is marginally stable in the directions of the vanishing eigenvectors of the Hessian. In order to stop a drift of the state along these directions due to noise an auxiliary stationary stimulus is added which has the form

$$s_{\mathrm{aux}}(dp, \theta) = \eta_{aux} \|dp\| \quad (27)$$

η_{aux} is a negative constant fixed according to experiments.

Another contribution to the stimulus can incorporate knowledge about the egomotion of the robot if available. It contributes to a disambiguation of the correlation information. For the important case of looming flow which results from a movement into the direction of the optical axis (see Section 3) this contribution takes the form

$$s_{\mathrm{egomotion}}(dp, \theta) = \begin{cases} 0 & : \; p + dp \in Op \\ \eta_{\mathrm{egomotion}} & : \; p + dp \notin Op \end{cases} \quad (28)$$

where Op is the straight line from the origin O of the image to the pixel p under consideration.

2.5 *Experimental comparison of the dynamic flow computation with the region-based correlational flow algorithm of Little et al.*

The evaluation of an optical flow algorithm should be done with respect to the purpose it has to subserve. It is not sensible to check simply the accuracy against some "veridic flow" since according to the behavior it has to subserve certain systematic errors are tolerable. In the context of obstacle avoidance on has to check against three criteria: (1) the density of the flow field (2) the accuracy of the flow field in particular: the absence of ghostmatches, and the detection of small obstacles (3) the computational load.

The evaluation of the dynamical flow computation is done by comparing it to the original region-based matching algorithm proposed by Little et al. This can serve to the reader interested in other flow field algorithms as a starting point since in the literature comparisons of the Little algorithm to a multitude of other approaches can be found (see e.g. Little et al. 1989, Bohrer 1993, Barron et al. 1994)

2.5.1 *Comparison of the density and the accuracy*

Fig.4 shows the results of dynamical flow computation. The images a) to c) show the evolution of the flow vectors over the last three steps of a sequence of five images acquired in our laboratory. Image d) shows as a comparison the result obtained with the original flow algorithm proposed by Little et al. First one should note that outliers with a too large amplitude do not appear for the dynamic flow field computation while for the original algorithm they are abundant (denoted by the little arrows). As already mentioned before a discrimination has to be made between matches that are an overestimate of the image velocity and matches which are an understimate of the image velocity. The first are usually more severe since a large image velocity is treated as being a result of a close object requiring immediate action. For example it might lead to an unneccesary turning manoeuvre. In contrary optical flow vectors with a too low amplitude are less severe given that the retinal size of the object encompasses several pixels. For safe navigation behavior it is in general sufficient if one pixel in a little neighborhood has a correct amplitude. Hence the strong reduction of the number of outliers facilitates the motion planning significantly.

2.5.2 *Comparison of the runtimes*

Among the "classical" flow field algorithms the one of Little is the fastest one. Two different implementations of the Little algorithm have been proposed which differ in their runtime behavior. The naive unoptimized version scales as follows (Bohrer 1993)

$$C_{\mathrm{Little}} = (2\delta + 1)^2((2\epsilon + 1)^2 + 1)HV \quad (29)$$

Hereby is $2\delta + 1$ the diameter of the searchwindow, $2\epsilon + 1$ is the diameter of the test window and H and V are the

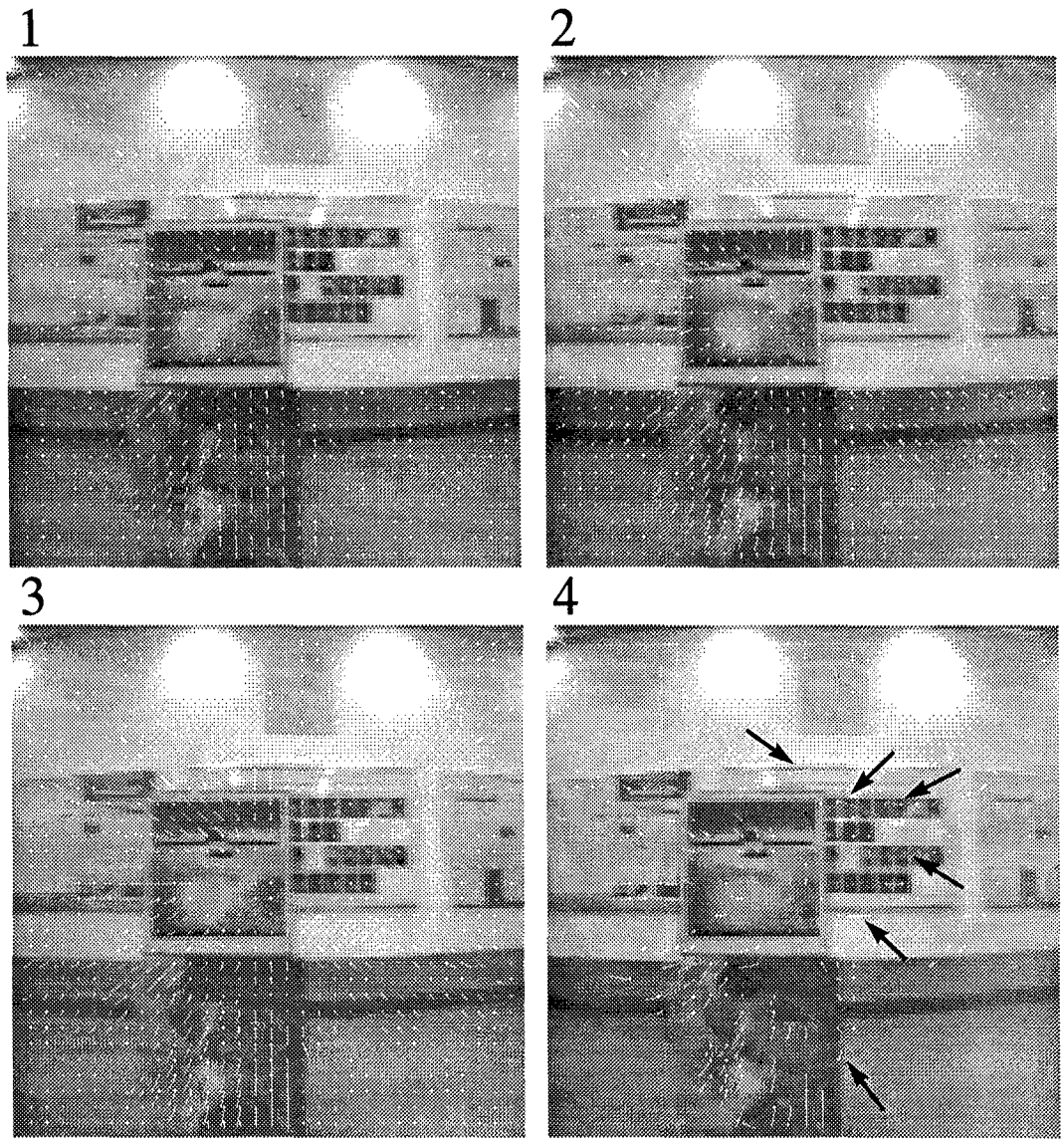


Figure 4: Flow vectors estimated using dynamical systems are shown. Figures 1) to 3) show their evolution over the last three timesteps in a sequence of five. For comparison the flow obtained with the original flow algorithm of Little et al. applied to the first and the fifth image is shown in Fig. 4). Note the outliers that are marked with the arrows. These occur especially in image regions with periodical structures or in image regions that show a grey value variation only in one direction. Correlation surfaces for which the best and the second best correlation score do not differ by a fixed difference are discarded from the flow computation. Increasing this difference eliminates errors at the price of a lower density of the flow field. (The image size is 128×128 pixels, the testwindow size is 5×5 pixels.)

horizontal and vertical image dimensions. In the case that for each pixel the size of the searchwindow as well as of the testwindow is equal one can optimize the algorithm by exploiting the fact that for neighboring pixels the testwindows are overlapping. Accordingly a part of the results of the computations necessary to determine the correlation surface for one pixel can be transferred to the neighboring pixel. This leads to the scaling

$$C_{\text{Little,optimized}} = (2\delta + 1)^2(4HV - H) \qquad (30)$$

The number of steps necessary for the dynamic computation of optical flow is given by

$$C_{\text{dyn}} = (2\epsilon + 1)^2(2\gamma + 1)^2 t_{\max} HV \qquad (31)$$

This holds if the correlation surface is not determined fully but only computed in a neighborhood of diameter $2\gamma + 1$ around the previous state $\mathcal{N}(dp_{\text{prev}})$. This is permissible since the acquisition parameters are chosen such that the maximum moves only less than one pixel per acquisition step. In this case it is also permissible to

replace the integration by a maximum detection in the neighborhood $\mathcal{N}(dp_{\text{prev}})$. Taking into account the fact that flow vectors at neighboring pixels tend to be similar because their testwindows overlap we compute flow only for a grid with a spacing that equals the diameter of the testwindow. This eliminates the factor $(2\epsilon+1)^2$. Inserting typical values ($H = V = 128, \gamma = 2, \delta = \Theta_{\max} = 4$) then shows that the dynamical flow computation is even faster than the optimized version of Little et al..

3 Motion planning with dynamical systems

A basic capability of a mobile robot is to move toward targets while avoiding obstacles. In the context of vision-based robots the problem has to be addressed that visual information tends to be very noisy. As a result decison making based on visual information becomes a non-trivial problem. Here we demonstrate how the dynamic property hysteresis helps to stabilize decisions such that coherent behavior results.

As a cue for collision danger we exploit the radial optical flows that result from rectilinear robot motion in the direction of the optical axis (Pichon et al. 1989, Coombs and Roberts 1992, Sandini et al. 1993). To this end the motion planning is done such that within between two turns of the robot a rectilinear movement is executed. During this forward motion two streams of images are acquired with the left and the right camera. The image streams are used to compute optical flow as outlined in the previous section 2. These flow fields, also called looming flow fields (see Figure 3), allow for estimating time-to-contact (Lee 1980, Gibson 1986, Horn 1986, Raviv and Herman 1993), in our case separately for the left and the right camera. For constant velocity rectilinear motion time-to-contact predicts the time to contact of the image plane with the obstacle which gives rise to the corresponding looming flow, hence the name.

Time-to-contact (ttc) is defined formally as $ttc(t) = d_{\text{obs}}(t)/v(t)$. Here $d_{\text{obs}}(t)$ is the projection of an obstacle's position onto the optical axis. The velocity, $v(t)$, is the obstacle's instantaneous velocity in a frame of reference attached to the camera. The estimation of time-to-contact from the radial optical flow makes use of the equation

$$ttc(t,p) = \frac{\|p\|}{\|dp(t)\|} \tag{32}$$

Here, $\|\cdot\|$ is the Euclidian vector norm. A time-to-contact is estimated for every pixel, p. Thus we have an entire time-to-contact map at our disposal. To simplify we consider for each camera only the minimal time-to-contact measured over the entire image

$$ttc(t) = \min_p\{ttc(t,p)\} \tag{33}$$

The most severe problem when working with time-to-contact is that the smallest time-to-contact measured over the whole visual field of one camera dominates the behavior. A single time-to-contact estimate is, however, fully affected by the noise on the estimates of the flow vectors and thus fluctuates strongly. Spatial averaging of the time-to-contact measurements is not an applicable solution to this problem. First, flow vectors measured for neighboring pixels are strongly correlated since their respective test windows overlap. Thus neighborhoods provide a poor ensemble for determining means. Moreover, and more severely, spatial averaging tends to swamp small obstacles in front of a far away background.

Our solution is to feed time-to-contact information into dynamical systems, one controlling the forward velocity and one the angular velocity (rotation rate). The forward velocity dynamics is designed to keep time-to-contact within a fixed range of values so that action planning can occur at a fixed time scale. The angular velocity dynamics in effect reacts to time-averaged time-to-contact estimates. This is both because it carries state and even inertia (it being a dynamics of rotation rate rather than heading direction) and because its nonlinear dynamics undergo hysteresis when turning directions are changed. These decisions themselves take the form of instabilities. This mechanism reconciles the need for flexibility with the need for short trajectories. A similar but more involved motion planning using dynamical sytems is discussed elsewhere (Neven and Schöner 1995, Neven and Schöner 1996). It is adapted to the original optical flow computation of Little et al.. These references also discuss visual guided target acquisition. Here advantage is taken of the superior quality of the optical flow determined via dynamical systems and accordingly emphasis is on how obstacles elicit avoidance behavior.

Therefore the planning of the forward velocity is kept trivial

$$v = \min\{\frac{d_{\text{goal}}}{\tau_{\text{goal}}}, \frac{ttcl}{TTC}v_{prev}, \frac{ttcr}{TTC}v_{prev}\} \tag{34}$$

Hereby d_{goal} is the current distance to the goal and τ_{goal} is a constant. The rationale behind this choice is on the one hand to stop at the goal position and on the other hand to keep the minimal time-to-contact approximately at the constant value TTC. This facilitates the flow computation since in this case the flow vectors do not exceed a maximal length.

Figure 5 shows the temporal evolution of the complete stimulus, $s_{\omega,\text{goal}} + s_{\omega,\text{obs}}$, along a typical path.

The angular velocity is generated from the dynamics

$$\dot{\omega} = \frac{1}{\tau_\omega}\int_{-\infty}^{\infty}(\omega'-\omega)\exp\left(-\frac{(\omega'-\omega)^2}{c_\omega^2}\right)\left(s_{\omega,\text{goal}}(\omega',t) + s_{\omega,\text{obs}}(\omega',t)\right)d\omega' \tag{35}$$

The contributions to the dynamics reflecting target acquisition (index goal) and obstacle avoidance (index obs)

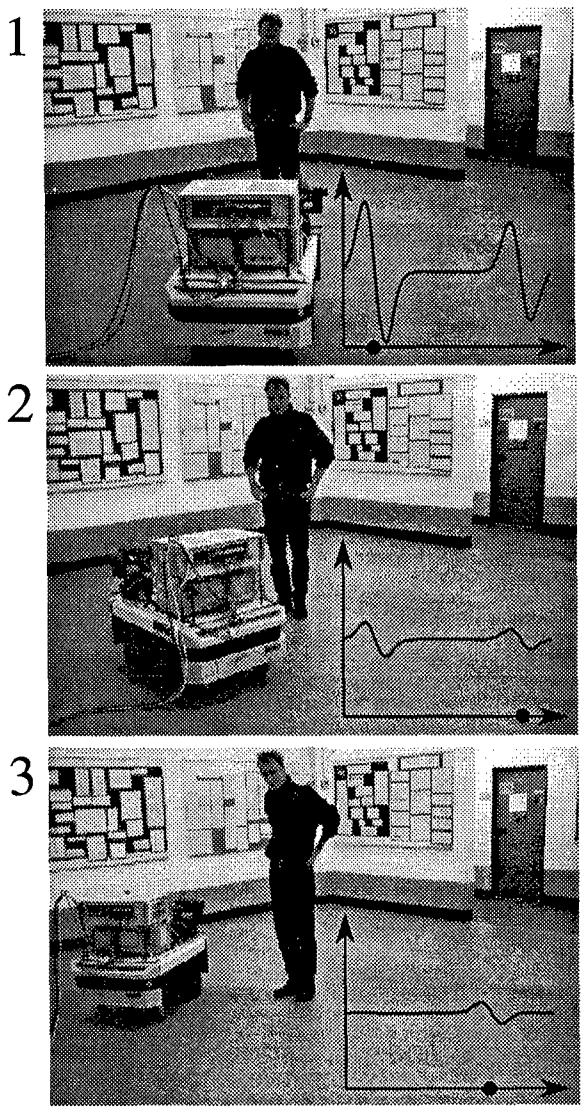


Figure 5: The robot heads to a goal while it avoids an obstacle. The insets show the phaseplot of the dynamics which manages the angular velocity of the robot. The black dot denotes the state of the dynamics. 1) Two strong obstacle contributions make the potential bistable. 2) Due to hysteresis the robot keeps on turning left towards a goal position despite a spurious obstacle entry on the left. 3) Only the goal contribution acts on the dynamics. Thus it is effectively reduced to a proportional controller. (In 2) and 3) the goal contributions are not to scale.)

are added. The idea of how obstacles elicit avoidance behavior is this: Add two symmetric terms to the rotation rate dynamics, one specifying rotation to the left at maximal rate, the other rotation to the right at maximal rate. The strengths of these two contributions depend on the time-to-contact as determined by the right camera and by the left camera, respectively. Whichever side has a smaller time-to-contact, strengthens the contribution for turning to the contra-lateral side, so that a tendency is created to turn away from the side with smaller time-to-contact. This is much in the spirit of Braitenberg's hypothetical vehicle number two (Braitenberg, 1984), a purely behavior-based robot in modern jargon. Imposing this control not directly onto the motor command, but onto a dynamics of the motor command, makes this a stable control strategy: Over a wide range of time-to-contact differences between left and right the resultant dynamics of rotation rate is bistable. Hysteresis makes sure that as the form of the contributions change due to fluctuations and changing sensory information while driving, a decision to turn either left and right is maintained, until overwhelming evidence for the opposite decision has been acquired. At this point an instability brings about a behavioral switch. Given the very noisy time-to-contact measurements, this form of stabilization of the vehicle trajectory and the underlying decisions through hysteresis is indispensable. Were one, for instance, to replace the behavioral dynamics by an algorithm that communicates at each instance the direction with maximum stimulus to the vehicle, the robot would easily be trapped in oscillatory movements as we were able to demonstrate empirically. Mathematically,

$$s_{\omega,\mathrm{obs}}(\omega) = \eta_{\omega,\mathrm{obs}}\left(\mathrm{ttcl}(t)\right)\delta(\omega-\omega_{\max}) + \eta_{\omega,\mathrm{obs}}\left(ttcr(t)\right)\delta(\omega+\omega_{\max}) \tag{36}$$

The coupling strength, $\eta_{\omega,\mathrm{obs}}(\mathrm{ttc.})$ is chosen as

$$\eta_{\omega,\mathrm{obs}}(ttc.(t)) = s_1\,\xi\left(v_{\mathrm{prev}}\mathrm{ttc.}(t) - D_1\right) \tag{37}$$

where s_1 is a negative, D_1 a positive constant and $\xi()$ is the function defined as

$$\xi(a) = \begin{cases} a & : \quad a \geq 0 \\ 0 & : \quad \text{else} \end{cases} \tag{38}$$

The goal contribution is given by

$$s_{\omega,\mathrm{goal}}(\omega) = \eta_{\omega,\mathrm{goal}}\delta(\omega - \frac{\phi_{\mathrm{goal}}}{\tau_{\mathrm{goal}}}) \tag{39}$$

Hereby is $\eta_{\omega,\mathrm{goal}}$ a constant and ϕ_{goal} is the current goal direction.

4 Conclusion

In this article we described further applications of the dynamic approach to autonomous systems design. We demonstrated that interpreting correlation surfaces as potentials for dynamical systems whose internal states estimate the flow vectors renders optical flow more useful for motion planning. The scaling behavior of dynamical flow computation is such that for many practical cases the computational load is comparable to the fast correlational algorithm of Little et al.

Motion planning using dynamical systems also proved useful in connection with the new type of flow computation. A little disappointment was to see that the stop and shot image sequences taken by our robot do sometimes not fullfill the restriction that for dynamical flow computation image velocity should be less than one pixel per frame. This is because the robot movements were too jerky to allow for such an acquisition mode in discrete steps. Rather the image acquisition has to proceed in a fashion that ensures small image displacements between successive frames which is possible if more advanced image processing hardware is employed. Nevertheless did the hysteresis property of the angular velocity dynamics ensure that a smooth trajectory comes about despite occasional erroneous time-to-contact estimates.

Acknowledgments

We are indepted to G Schöner who developped the dynamic approach to autonomous systems design together with us. The technical assistance of M Neef and R Menzner is acknowledged. We thank C von der Malsburg and W von Seelen along with their collaborators for inspiring discussions. P Paschke helped us correcting the manuscript.

References

[1] R C Arkin. Integrating behavioral, perceptual, and world knowledge in reactive navigation. *Robotics and autonomous control*, 6:105–122, 1990.

[2] E Bicho, G Schöner The dynamic approach to autonomous robotics demonstrated on a low-level vehicle platform, accepted for SIRS96.

[3] S Bohrer, HH Bülthoff, and HA Mallot. Motion detection by correlation and voting. In *Parallel Processing in Neural Systems and Computers*, pages 471–474. Elsevier Science Publishers B.V., North-Holland, 1990.

[4] HH Bülthoff, JJ Little, and T Poggio. A parallel algorithm for real-time computation of optical flow. *Nature*, 337 (9), pages 543–553, 1989.

[5] Coombs D and Roberts K. "Bee-bot": using peripheral optical flow to avoid obstacles. In *spie-robots 92*, 1992.

[6] JJ Gibson. *The Ecological Approach to Visual Perception.* Lawrence Erlbaum Associates, 1986.

[7] RM Haralick and LG Shapiro. *Robot and Computer Vision, Volumes 1 and 2.* Addison-Wesley, Redwood City CA, 1992.

[8] BKP Horn. *Robot Vision.* The MIT Press, Cambridge MA, 1986.

[9] O Holland. New robotics and neural networks In *Proceedings ICANN '95, Industrial Conference.*

[10] H Neven and G Schöner. Behavior-Based Visual Navigation: Theory and Experiment in *A Neural Architecture for Autonomous Visually Guided Robots - Results of the* NAMOS *Project* - Fortschritt-Berichte VDI, W. von Seelen (ed.), 1995.

[11] H Neven and G Schöner. Dynamics parametrically controlled by image correlations organize robot navigation. *BiolCyb*, in press.

[12] Lee DN. Visuo-motor coordination in space-time. In *Tutorials in Motor Behavior*, pages 281–293. eds. G E Stelmach and J Requin, Amsterdam: North-Holland, 1980.

[13] JM Pichon, CH Blanes, and N Franceschini. Visual guidance of a mobile robot equipped with a network of self-motion sensors. In *SPIE Vol. 1195 Mobile Robots IV*, pages 44–53, 1989.

[14] D Raviv and M Herman. Visual servoing from 2-d image cues. In Y Aloimonos, editor, *Active Perception*, pages 191–226. Lawrence Erlbaum Associates, 1993.

[15] G Sandini, F Gandolfo, E Grosso, and M Tistarelli. Vision during action. In *Active-Perception*, chapter 4, pages 151–190. Lawrence Erlbaum Associates, Hillsdale (NJ), Hove and London, 1993.

[16] G Schöner and M Dose. A dynamical systems approach to task-level system integration used to plan and control autonomous vehicle motion. *Robotics and Autonomous Systems*, 10:253–267, 1992.

[17] G Schöner, M Dose, and C Engels. Dynamics of behavior: theory and applications for autonomous robot architectures. *Robotics and Autonomous Systems*, submitted.

[18] A Steinhage and G Schöner. Self-calibration based on invariant view recognition: Dynamic approach to navigation, *Robotics and Autonomous Systems* in press.

Postural primitives: Interactive Behavior for a Humanoid Robot Arm

Matthew M. Williamson*
matt@ai.mit.edu
MIT Artificial Intelligence Lab,
545 Technology Square, Room NE43-828, Cambridge, MA 02139

Abstract

This paper describes the implementation of reflex action for the arm of the humanoid robot Cog [5]. A set of biologically inspired postural primitives are used to create the arm motion. The primitives are combined in different ways to achieve reaching with grasping and withdrawal reflexes, allowing the arm to interact safely with both objects and people. This paper describes the reflexes, the biological inspiration for the control, and includes data collected from the robot.

1 Introduction

The humanoid robot Cog [5] is intended to explore its environment using its body. There are many possible ways for Cog's arms to perform this role, and this paper describes the first stage—reaching with reflexes. This allows the arm to move around safely and interact with objects and people.

Specifically, the arm reaches from a rest position to a random target (this has recently been extended to targets in visual coordinates, Marjanović at al. [18]). It will grasp—the arm stopping whenever something touches the palm, and withdraw—returning to the rest position if the top of the hand collides with anything. Also implemented is compliant motion, or a "lead" behavior, allowing the arm to be lead around to a new position, if the hand is held.

Later stages, which will lead to greater accuracy in reaching, and compensation for dynamics, are intended to be layered on top of this low-level system.

The control system used to implement this behavior has a number of different levels, both of hardware and software, and is heavily based on biological models of movement control. The arm controller design has the following biologically inspired features.

- It uses reflexes that appear in developmental stages of children (Diamond [9]).
- The arm joints have a spring-like behavior similar to that of humans.
- It controls motion using a set of postural primitives, similar to those observed in frogs and rats (Bizzi [2]).
- Conflicts and interactions between primitives are resolved using both superposition and winner-take-all, which has also been observed by Mussa-Ivaldi [23]
- It sums motion trajectories to achieve smooth motion.

The remaining sections of this paper describe in detail the biological inspiration for the arm control, the reflex network responsible for the arm behaviour, and the hardware implementation of the system. Conclusions and references to previous work are included towards the end of the paper.

2 Biological Basis

2.1 Springy Joints

It is intended for Cog's arm to be human-like, which means at least having a spring-like behaviour. There is a considerable amount of biological evidence for the spring-like behaviour of muscles (See Zajac [33] for review) and also for the spring/damper-like behaviour of individual joints (Cannon and Zahalak [6], MacKay et al. [17]). There is evidence that joints act like springs of constant stiffness, from measurements of the end-point impedance of human arms (Mussa-Ivaldi [24]). By changing the stiffness of the various muscles, the impedance (springiness) of the hand can be changed which may help to perform different tasks (Hogan [14]). In an attempt to mimic this, the actuators of Cog's arm behave as if they are springs of variable spring-rate and damping. Motion is achieved by changing the rest or equilibrium position of the springs θ_{set} as detailed in Figure 1. As shown in the figure, the torque applied to the joint τ can be written as

$$\tau = K_{joint}(\theta - \theta_{set}) + B_{joint}(\dot{\theta})$$

*Support for this research was provided JPL Contract # 959333

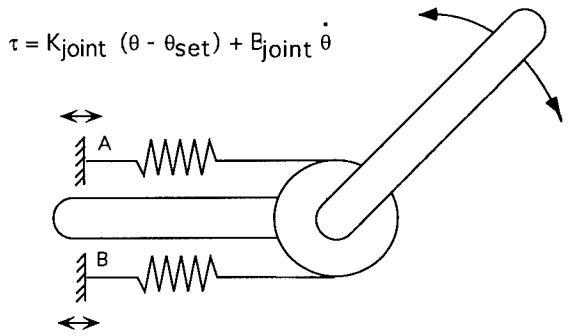


Figure 1: The spring-like actuators of Cog's arm. The diagram shows a reasonable model of the biological joint. The joint is moved by moving the end of the springs marked A and B in opposite directions . By changing the stiffnesses of the springs, the joint stiffness can be altered. The joint will have a natural equilibrium position (θ_{set}), but will generally be at a different angle (θ) due to gravitational and dynamic loads. If a damper is also included, the net torque on the joint can be given by the equation above. In Cog's arm a motor is used to produce this behavior.

This system has a sensible "natural" behaviour, if it is disturbed, or hits an obstacle, the arm simply deflects out of the way. The disturbance is dealt with by the characteristics of the system, and needs no explicit sensing or computation. Secondly since the system has a low frequency characteristic (large masses and soft springs), commands can be sent at a low rate, while still obtaining smooth arm motion. This allows more time for computation, and allows control from a system with long delays, perhaps more akin to biological systems. Thirdly, if the joint set-points are fed-forward to the arm, then the stability of the system is guaranteed.

A disadvantage of this kind of system is that it becomes more complicated to perform more traditional robotic tasks such as pure force control, (you would need to modify the set-points of the springs at a high rate) and also the low stiffness of the joints mediates against accurate position control. Humans generally achieve good position control by increasing stiffness (co-contraction) and by eliminating extra degrees of freedom (by bracing their hands while writing for example), and similar techniques would also be appropriate for Cog's arm.

2.2 *Motion Primitives*

Many researchers have suggested a hierarchical control scheme for movement control, based on arguments of bandwidth (there are many sensors and muscles which need a large amount of information to function correctly), delays (nerves are slow, and communicating all that information would result in long delays, with subsequent control problems), and anatomy (the loop from muscles to spinal cord to sensors suggests heavily some low level control modulated by descending signals from the brain). Experiments by Bizzi and others (Bizzi [2], Giszter [13], Mussa-Ivaldi [23], and Loeb [16]) have attempted to elucidate this hierarchy. They electrically stimulated the spinal cords of frogs and rats and measured the forces obtained at the leg, mapping out a force field in leg-motion space. This is shown in Figure 2.

Surprisingly they found that the fields were generally convergent (with a single equilibrium point) and uniform across different frogs. The idea is that if the leg is free to move, it will move to the center of the field, see Giszter [13]. Different fields therefore correspond to different postures of the leg in space. They have found only a small number of fields (4 in total), which correspond to postures either at the extremes of the workspace or to postures involved in reflex actions (such as wiping the legs against one-another).

They have found that the fields can be combined either by superposition—stimulating in two places in the spinal cord resulted in a field which was the linear vector superposition of the fields obtained when each point was stimulated, or by "winner-take-all"—stimulating in two places, and finding the resulting field coming from one of the two places (Mussa-Ivaldi [23]). They also obtained similar fields from chemical stimulation of the spinal cord, and from cutaneous simulation. The force fields remain of similar shape even after the frog is deafferented (Loeb [16]), which strongly suggests that the fields are caused by the interaction of the constant (there is no sensory feedback) spring-like properties of the muscles and the leg kinematics.

These findings lead the researchers to suggest that these fields are primitives that are combined to obtain leg motion. Mussa-Ivaldi et al. [22] have shown that fairly arbitrary force patterns, and so complex motions can be generated using a few of these force field primitives.

In Cog's arm the primitives are implemented as a set of set-points for each of the arm joints, see Figure 3. This holds the arm in a position in space. If the arm is deflected, then there is a force moving the hand back to the equilibrium position, from the action of the springy joints. A primitive is defined as a vector $P_i = (\theta_{set1}, \theta_{set2} \ldots \theta_{set6})'$ (for an arm with 6 degrees of freedom). The position of the end of the arm is generally given by the forward kinematics (see any robotics textbook such as Paul [28]). This relates the position of the end of the arm $\vec{X}$ to the joint angles θ.

$$\vec{X} = L(\theta)$$

Cog's arm is springy, so the actual joint angles are not the same as the set-point or equilibrium angles. They are related through the dynamics of the whole system

$$M(\theta)\ddot{\theta} + C(\theta, \dot{\theta}) + G(\theta) = K_{joint}(\theta - \theta_{set}) + B_{joint}\dot{\theta}$$

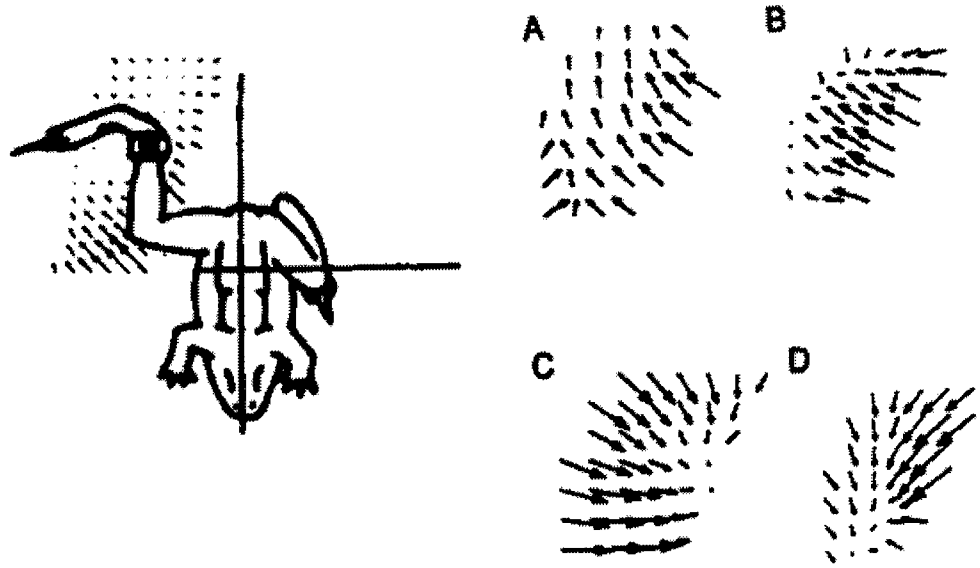


Figure 2: Convergent force fields in a frog's spinal cord. The left part of the figure shows the frog with the force transducer on its ankle. A force field is depicted under the leg. The right hand part of the figure shows the four basis fields, which are **A** leg back, **B** leg out to the side, **C** leg tucked up against the body, and **D** leg forward. Reproduced from Mussa-Ivaldi [22], with permission.

Where M, C and G have the usual meanings of inertia, coriolis and gravity terms. The arm has one equilibrium point, so in general there is another forward kinematic relation between the endpoint position $\vec{X}$ and the set-points θset.

$$\vec{X} = L(\theta) = L'(\theta_{set}) = L'(P)$$

One way to combine the primitives is by winner-take-all which means that one of the P_i has control of the arm. Alternatively linear superposition can be used:

$$P = \alpha_1 P_1 + \alpha_2 P_2 + \cdots + \alpha_n P_n$$

P also specifies a posture in space, which will be somewhere between all the P_i's. If $\sum_0^n \alpha_i = 1$ then the region of space that the arm can move in is bounded. This arrangement allows the arm to move around in a bounded region, the corners of which are the primitives, as shown in Figure 3. Unfortunately, due to the non-linearity of the forward kinematics,

$$L'(P) \neq L'(\alpha_1 P_1) + L'(\alpha_2 P_2) + \cdots + L'(\alpha_n P_n)$$

so interpolation in primitive space does not exactly match interpolation in Cartesian space. The complexity of this mapping can be greatly affected by particular choice of primitive vectors.

The force field $\vec{F}$ from a particular choice of posture is obtained by considering $\tau = J^T \vec{F}$, (see Paul [28]) and is given by the solution of

$$J(\theta)\vec{F} = K_{joint}(\theta - \theta set)$$

where $J(\theta)$ is the Jacobian from joint angles to Cartesian coordinates. This mapping produces a field with a non-linear shape.

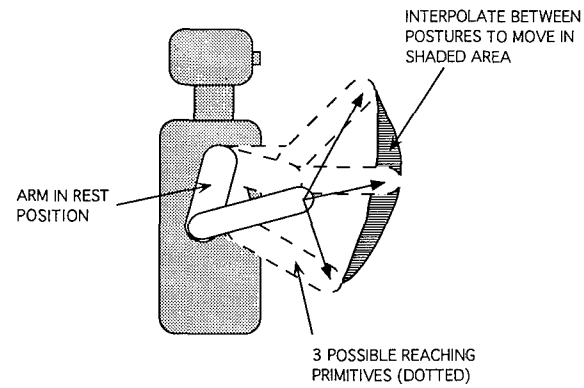


Figure 3: Primitives. Here four primitives are defined for a reaching task: a rest position, and three in front of the robot. Linear interpolation is used to reach to points in the shaded area. See also Figure 5.

The hierarchical organization has some advantages. There is a reduction in bandwidth as the commands to the arm need only set the rest positions of the springs, and do not deal with the torques directly. If a small number of primitives are used, there is also a reduction in the dimensionality of the system, with a corresponding reduction in complexity. The sacrifice made is that the arm becomes less redundant which may limit flexibility. The motion is bounded which is useful if there

are known obstacles (like the body of the robot!). The primitive framework provides a very clean way of implementing the kind of withdrawal and grasping reflexes which are described in more detail in the following sections. For example, to implement the withdrawal reflex, winner-take-all is used to move the arm to the rest posture. To reach to a target, interpolation is used to move the arm to the correct position.

2.3 Smoothness and trajectory combination

The previous sections described the implementation and combination of primitives in space, whereas it is a different issue how to combine them in time. Humans when performing reaching tasks tend to move the ends of their arms in roughly straight lines, with remarkably consistent bell-shaped velocity profiles. This seems to be independent of the region of the workspace (Morasso [20], Flash and Hogan [12], Hollerbach and Flash [15]). Slow movements do tend to be more curved however (Cruse and Brüwer [7]). There has been much argument about whether these motions are planned in Cartesian space, using an optimization such as minimum jerk (Nelson [26]), or in joint space using minimum joint torque change (Uno et al [31]). Independent of what is the exact criterion, it is clear that humans move their arms smoothly.

Whenever a reflex action is initiated, the target of the arm motion changes. When humans move between a number of targets (Flash [11], Flash and Henis [12]), or make a movement that requires precision (Milner [19]), they still exhibit smooth profiles, but seem to use a number of discrete smooth movements, rather than continuous motion. This has also been observed in infants (Hofsten [32]). One conclusion is that the trajectories are summed, so that the actual motion is the sum of the old motion and the motion needed to get from the old target to the new one (Flash [12]). An alternative model is that the segmentation is an artifact of the interaction between the change in the target position and the limb characteristics (Flanagan et al. [10]).

For Cog's arms, the motion is implemented by changing the set-points of the joints using a smooth minimum jerk profile (Nelson [26]). The velocity profile for each joint is calculated from

$$\dot{\theta}_{set} = \frac{\theta_{end} - \theta_{start}}{d} \left[30(t/d)^4 - 60(t/d)^3 + 30(t/d)^2\right]$$

where t is time, and d is the duration of the move. To combine trajectories, a summation strategy is implemented. The velocities for all the active trajectories are summed and integrated to calculate the command to the arm. This allows the target to change during the motion, while still obtaining smooth motion. The new motion is from the old target to the new target, and does not require knowledge of the arm position. This is illustrated in Figure 4. The velocity trace is qualitatively similar to those presented in [11, 12, 19].

Since the interpolation is carried out in joint space, the end-point of the arm does not travel in a straight line in Cartesian space. In addition the motion is created by commanding the set-points of the arm, while the actual arm motion is dependent on gravity and dynamical loads, which can make the actual arm trajectory less smooth. This is an area where further work is needed. It is interesting that the "straightness" of infant reaches increases, as they grow older (Hofsten [32]).

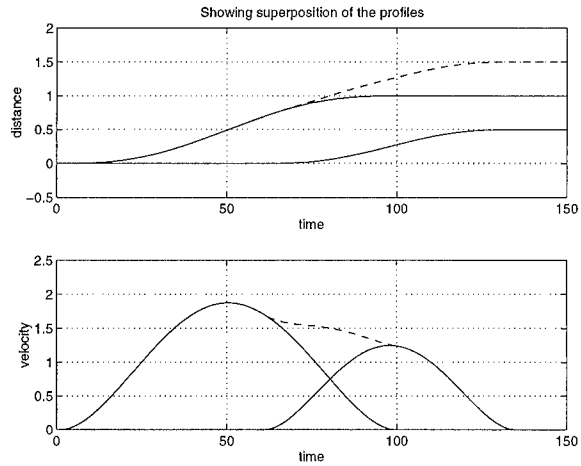


Figure 4: Implementation of summing of trajectories. The first move was from 0 to 1, then another move was added to 1.5. The top trace shows the two individual positions, and the dotted trace shows the actual motion. The lower graph shows the combination of the velocity profiles, again the dotted trace being the result of the combination.

3 Description of the reflex control

The arm is loosely based on the dimensions of a human arm, and is illustrated in Figure 5. It has 6 degrees of freedom, each powered by a DC electric motor with a series spring (Series Elastic Actuator, see Pratt and Williamson [29]). The spring is used to give good force control at the joint, and to protect the motor gearbox from shock loading. At the end of the arm is mounted a simple claw, which has touch sensors to detect when the arm is touched. These sensors then initiate the reflex responses.

The hardware setup is shown in Figure 6. Each motor is controlled using a dedicated Motorola 6811 microcontroller which runs a 1kHz control loop, creating the virtual spring behavior and managing the sensors. All the spring-like behaviour is implemented at this low level.

The 6811's communicate with a Motorola 68332 processor at about 50Hz, receiving joint set-points, control loop gains and spring parameters, and returning

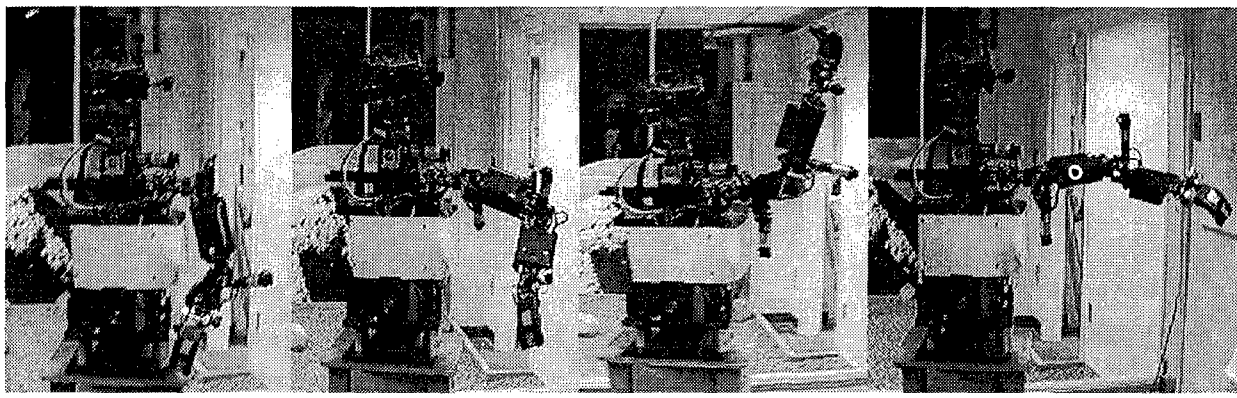


Figure 5: Picture of Cog and its arm. The four pictures also show four primitive postures, left to right, rest position, down, up and to the side. The claw at the end of the arm has 4 touch sensors, which are used by the reflexes.

torque, position, and other sensory information. The reflex network itself is implemented on another two 68332s, which communicate with each other through a shared dual-ported ram (dpram) memory interface. One of the 68332s generates targets for the reaching, and the other calculates trajectories, and deals with the sensory information. The trajectories are communicated by dpram to the communication 68332 and from there to the individual joint controllers. One final 68332 manages communication from the 68332s to a front end Macintosh computer.

The 68332s are programmed in L [4], a subset of Common LISP. This overall system architecture was chosen for Cog's brain to allow local functionality, but wide ranging communication between different parts of the brain. Indeed, this reflex system has been interfaced with the visual system getting targets for the arm motion from visual cues (Marjanović [18]).

The organization of the reflex network is shown in Figure 7. It is based on a subsumption architecture approach (Brooks [3]). The basic behavior is reaching, for which the target generator generates targets, which generate trajectories which move the arm. If the withdrawal reflex is initiated, from something touching the appropriate sensor, a new trajectory is planned to return the arm to the rest position. This new target suppresses any new targets from the target generator, and returns the arm to the rest posture. The grasp works in a similar way, adding a trajectory to return the arm to where it was first grasped, and also suppressing other targets. The lead behavior monitors the torque at each of the joints. If the behavior is initiated, it backs off the set-point of the joints if the torque goes above a threshold. Each joint is dealt with independently. It also suppresses any new targets from being processed. The trajectory generator is updated with the latest command position, so that reaching can continue as normal once the leading has finished.

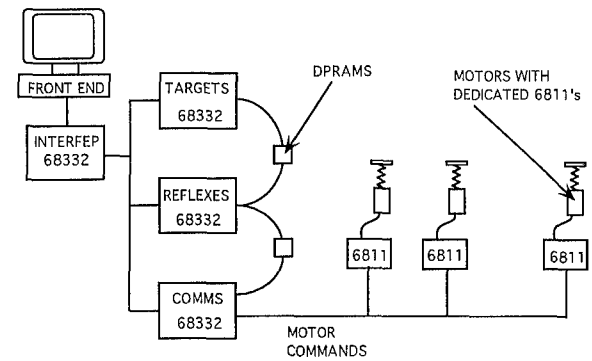


Figure 6: The arrangement of the hardware for the reaching network. Each joint has a dedicated 6811 which implements the spring-like behavior. A bank of 68332's is used to calculate targets and trajectories, hold the reflex network and communicate with the 6811's. One further 68832 manages communication between the processor bank and the Macintosh front end.

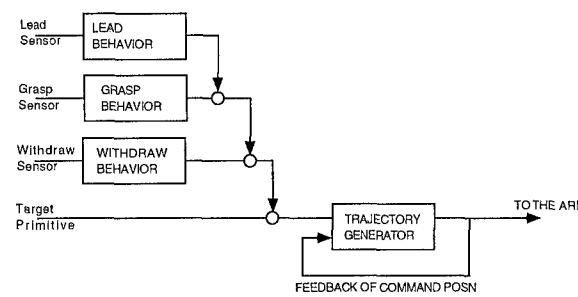


Figure 7: Figure showing the reflex network. The empty circles correspond to suppression of the lower wire.

4 Results

The arm control and reflex network worked well, and fulfilled the goal of allowing people to interact safely with the robot. The arm can be pushed around and deflected when it is stationary or moving, without causing damage.

In a complete system such as this it is difficult to find a format in which to present results and performance. This is compounded by there being no way at present to measure the position of the hand. In the results that follow, the effect of the reflex on one of the joints is described. In reality all 6 joints are moving, but the behavior of only one joint is shown for clarity.

Figure 8 shows the action of the grasp reflex. The graph shows the joint set-points, the actual joint positions and the reading on the touch sensor. The sinusoidal motion comes from the arm reaching out and returning to its rest position. The effect of the gravity loading is clear, since there is quite a large difference between the actual and the rest position of the joint. This difference also changes as the joint moves forward and up (positive on the graph) and then down again. When the grasp reflex is initiated, the arm quickly stops, and holds itself in the stopped position. The actual joint position and the set-points are different because of the gravity loading.

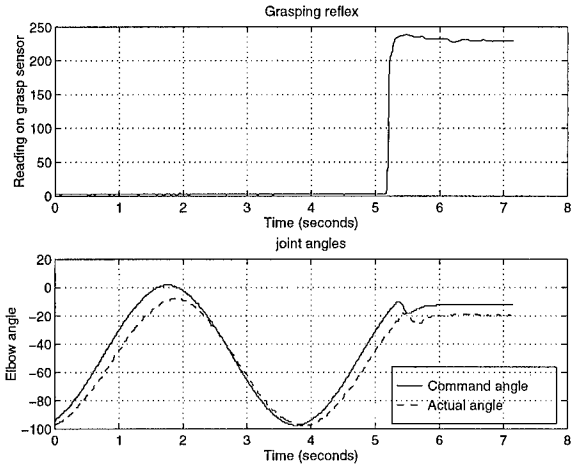


Figure 8: Grasping. The behavior of the shoulder joint is depicted in this graph. The sinusoidal motion comes from the arm reaching forward and back, until it is interrupted by the grasp reflex. The top trace shows the sensor reading, and the bottom trace show the actual (dashed) and equilibrium (solid) position of the joint. The effect of the dynamical and gravity loading is clear from the difference between the curves in the bottom trace. When the grasp is initiated, the arm quickly stops, and stays where it was originally touched.

Figure 9 shows a similar picture for the withdrawal reflex. Again the arm is moving forward and back, and when the withdraw sensor is touched, a fast trajectory back to the rest position is performed. The arm stays in the rest position for a short time, until a new target arrives from another part of the system.

Figure 10 shows the results for the leading behavior. When the hand is held, the arm is lead to a new position. This is accomplished by monitoring the torque at all of the joints, and backing off the joint equilibrium positions to keep the torque low. When leading is started, the system records the torque at all the joints. If, during the subsequent motion, the torque changes very much from that initial value, the set-point of the joints is altered accordingly. In order to compensate for the different loads that will be experienced as the arm moves through different postures, the initial torque reading is updated at a slow rate. The top graph shows the torques of the shoulder joint, as well as the slowly updating initial torque value. The bottom trace show the motion of the joint, and the changes in the set-point as the torque becomes too high. In the middle of the trace, the arm was released, and moved freely, the set-point remaining constant. The reaching behaviour was reinitiated by touching the sensor again at the end of the trace.

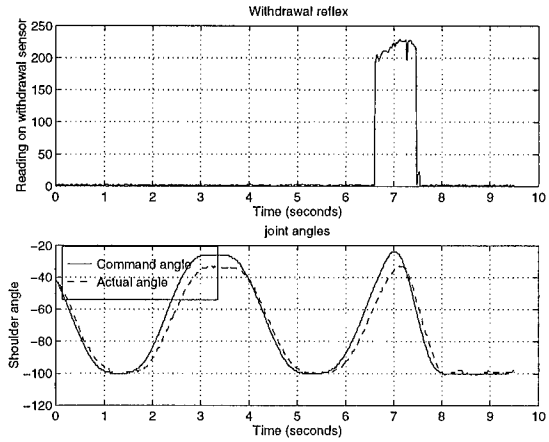


Figure 9: Withdrawal. The behavior of the shoulder joint is depicted in this graph. The sinusoidal motion comes from the arm reaching forward and back until it is interrupted by the withdrawal reflex. The top trace shows the sensor reading, and the bottom trace show the actual (dashed) and equilibrium (solid) position of the joint. When the withdraw is initiated by the high reading on the touch sensor, the arm quickly returns to the rest position.

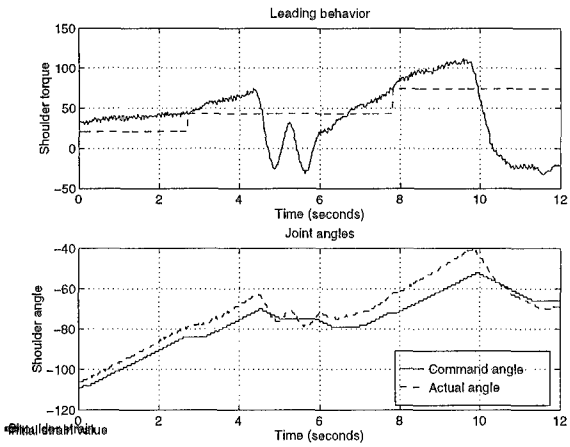


Figure 10: Leading. The graph shows the behavior of the shoulder joint as the arm is moved. The top trace shows the torque on the shoulder joint, as well as the value of the initial torque. The torque (solid line) is larger than the initial torque (dashed line) so the command angle or set-point of the joint is changed (solid line in lower graph). The initial torque reading is updated every 5 seconds to allow for changes in the torque due to postural changes. It is updated to the current torque reading (at about 3 secs, and at 8 secs). The bottom trace shows the movement of the arm (dashed) as well as the set-point (solid). In the middle of the trace (between 5 and 6 sec), the arm is released, and the leading is turned off. The set-point remains constant, and the arm can be moved without affecting the set-point (wiggles on torque and actual angle traces). At 6 seconds, the leading behavior is re-initiated, with the set-point changing as the torque changes.

5 Conclusions

This paper describes the implementation of a controller for the motion of Cog's arm. It uses sensory information to determine the motion of the whole arm. There have been fairly few approaches that have followed this path. One example is Asteroth et al. [1], who used sensory information to get a robot arm to track a moving ball. Most uses of tight sensor-motor loops in robotics have been to cope with local uncertainty during assembly, the behavior-based part commanded by a high level planner (Smithers and Malcolm [30], Morrow et al. [21], Paetsch et al. [27]). Other workers have suggested the use of hierarchical controllers and primitives for whole arm motion (Deno et al. [8], Nelson et al. [25]). The approach taken in this paper differs in that the sensory information is used to affect the overall behavior of the arm, not just modulating the planned motion of the robot end-effector.

The approach described in this paper works well, giving robust behavior to the whole arm. At the lowest level, the spring-like behavior of the joints deals cleanly with unexpected collisions, and makes the arm stable. The postural primitives which are implemented on top of that behavior make the implementation of reflexes such as withdrawal and grasping easy. The whole system is safe and certainly achieves its goal of interactability.

There are two areas which need further work, the first being compensation for dynamical loading, which will make the arm motion straighter and smoother, and the second being layering more complex motions on top of this reflex base. More complexity will need the definition of more primitives, and also the use of vision and finer touch sensing.

References

[1] Alexander Asteroth, Mark S. Fischer, Knut Möller, and Uwe Schnepf. Tracking and grasping of moving objects - a behaviour-based approach. In *Proceedings of the 5th Intl Conf IEA/AIE-92. Industrial and Engineering Applications of AI and Expert systems*, volume 1, pages 195–204, 1992.

[2] Emilio Bizzi, Ferdinando A. Mussa-Ivaldi, and Simon F. Giszter. Computations underlying the execution of movement: A biological perspective. *Science*, 253:287–291, 1991.

[3] Rodney A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, RA-2:14–23, April 1986.

[4] Rodney A. Brooks. *The L Manual.* IS Robotics, Cambridge MA, 1995.

[5] Rodney A. Brooks and Lynn Andrea Stein. Building brains for bodies. *Autonomous Robots*, 1:7–25, 1994.

[6] S. Cannon and G. I. Zahalak. The mechanical behavior of active human skeletal muscle in small oscillations. *Journal of Biomechanics*, 15:111–121, 1982.

[7] H. Cruse and M. Brüwer. The human arm as a redundant manipulator:the control of path and joint angles. *Biological Cybernetics*, 57:137–144, 1987.

[8] D. Curtis Deno, Richard M. Murray, Kristofer S. J. Pister, and S. Shankar Sastry. Control primitives for robot systems. In *Proceedings of the 1990 IEEE Intl Conf on Robotics and Automation*, pages 1866–1871, Cincinnati, OH, 1990.

[9] Adele Diamond. Developmental time course in human infants and infant monkeys, and the neural bases of, inhibitory control in reaching. *Annals of the New York Academy of Sciences*, 608:637–676, 1990.

[10] J. Randall Flanagan, David J. Ostry, and Anatol G. Feldman. Control of trajectory modifications in

target-directed reaching. *Journal of Motor Behavior*, 25(3):140–152, 1993.

[11] Tamar Flash and Ealan Henis. Arm trajectory modifications during reaching towards visual targets. *Journal of Cognitive Neuroscience*, 3(3):220–230, 1991.

[12] Tamar Flash and Neville Hogan. The coordination of arm movements:an experimentally confirmed mathematical model. *Journal of Neuroscience*, 5(7):1688–1703, July 1985.

[13] Simon F. Giszter, Ferdinando A. Mussa-Ivaldi, and Emilio Bizzi. Convergent force fields organized in the frog's spinal cord. *Journal of Neuroscience*, 13(2):467–491, 1993.

[14] Neville Hogan. The mechanics of multi-joint posture and movement control. *Biological Cybernetics*, 52:315–331, 1985.

[15] John M. Hollerbach and Tamar Flash. Dynamic interactions between limb segments during planar arm movement. *Biological Cybernetics*, 44:67–77, 1982.

[16] Eric P. Loeb, Simon F. Giszter, P. Borghesani, and Emilio Bizzi. Effects of dorsal root cut on the forces evoked by spinal microstimulation in the spinalized frog. *Somatosensory and Motor Research*, 13(2):467–491, 1993.

[17] W. A. MacKay, D. J. Crammond, H. C. Kwan, and J. T. Murphy. Measurements of human forearm posture viscoelasticity. *Journal of Biomechanics*, 19:231–238, 1986.

[18] Matthew J. Marjanović, Brian Scassallati, and Matthew M. Williamson. Self-taught visually-guided pointing for a humanoid robot. Society of Adaptive Behavior, 1996. In these proceedings.

[19] T. E. Milner. A model for the generation of movements requiring endpoint precision. *Neuroscience*, 49(2):487–496, 1992.

[20] P. Morasso. Spatial control of arm movements. *Experimental Brain Research*, 42:223–227, 1981.

[21] J. Dan Morrow, Brad. J. Nelson, and Pradeep K. Khosla. Vision and force driven sensorimotor primitives for robotic assembly skills. In *Proceedings of the IEEE International Workshop on Intelligent Robots and Systems (IROS-95)*, volume 3, pages 234–240, Pittsburg, PA, July 1995.

[22] Ferdinando A. Mussa-Ivaldi and Simon F. Giszter. Vector field approximation: a computational paradigm for motor control and learning. *Biological Cybernetics*, 67:491–500, 1992.

[23] Ferdinando A. Mussa-Ivaldi, Simon F. Giszter, and Emilio Bizzi. Linear combinations of primitives in vertebrate motor control. *Proceedings of the National Academy of Sciences*, 91:7534–7538, August 1994.

[24] Ferdinando A. Mussa-Ivaldi, Neville Hogan, and Emilio Bizzi. Neural, mechanical, and geometric factors subserving arm posture in humans. *Journal of Neuroscience*, 5(10):2732–2743, October 1985.

[25] R. C. Nelson, M. Jägersand, and O. Fuentes. Virtual tools: A framework for simplifying sensory-motor control in complex robotic systems. In *Proceedings of the IEEE International Workshop on Vision for Robots (IROS-95)*, pages 70–79, Pittsburg, PA, July 1995.

[26] W. L. Nelson. Physical principles for economies of skilled movements. *Biological Cybernetics*, 46:135–147, 1983.

[27] W. Paestch and G. von Wichert. Solving insertion tasks with a multifingered gripper by fumbling. In *Proceedings of the 1993 IEEE Intl Conf on Robotics and Automation*, pages 173–179, Atlanta, GA, 1993.

[28] Richard P. Paul. *Robot Manipulators: Mathematics, Programming and Control, The Computer Control of Robot Manipulators*. MIT Press, Cambridge, Massachusetts, 1981.

[29] Gill A. Pratt and Matthew M. Williamson. Series elastic actuators. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS-95)*, volume 1, pages 399–406, Pittsburg, PA, July 1995.

[30] Tim Smithers and C. Malcolm. Programming robotic assembly in terms of task achieving behavioural modules. DAI Research Paper 417, University of Edinburgh, Edinburgh, Scotland, December 1988.

[31] Yoji Uno, M. Kawato, and R. Suzuki. Formation and control of optimal trajectory in human multijoint arm movement. *Biological Cybernetics*, 61:89–101, 1989.

[32] Claes von Hofstein. Structuring of early reaching movements: A longitudinal study. *Journal of Motor Behavior*, 23(4):280–292, 1991.

[33] Felix E. Zajac. Muscle and tendon:properties, models, scaling, and application to biomechanics and motor control. *CRC Critical Reviews of Biomedical Engineering*, 17(4):359–411, 1989.

ACTION SELECTION AND BEHAVIORAL SEQUENCES

Action Selection methods using Reinforcement Learning

Mark Humphrys
University of Cambridge, Computer Laboratory
New Museums Site, Cambridge CB2 3QG, England
Mark.Humphrys@cl.cam.ac.uk

http://www.cl.cam.ac.uk/users/mh10006/

Abstract

Action Selection schemes, when translated into precise algorithms, typically involve considerable design effort and tuning of parameters. Little work has been done on solving the problem using learning. This paper compares eight different methods of solving the action selection problem using Reinforcement Learning (learning from rewards). The methods range from centralised and cooperative to decentralised and selfish. They are tested in an artificial world and their performance, memory requirements and reactiveness are compared. Finally, the possibility of more exotic, ecosystem-like decentralised models are considered.

1 Action Selection

By Action Selection we do not mean the low-level problem of choice of action in pursuit of a single coherent goal. Rather we mean the higher-level problem of choice between conflicting and heterogenous goals. These goals are pursued in parallel. They *may* sometimes combine to achieve larger-scale goals, but in general they simply interfere with each other. They may not have any terminating conditions.

Typically, the action selection models proposed in ethology are not detailed enough to specify an algorithmic implementation (see [Tyrrell, 1993] for a survey, and for some difficulties in translating the conceptual models into computational ones). The models that do lend themselves to algorithmic implementation (e.g. see [Brooks, 1991, Blumberg, 1994, Sahota, 1994, Aylett, 1995]) then typically require a considerable design effort. In the literature, one sees formulas taking weighted sums of various quantities in an attempt to estimate the utility of actions. There is much hand-coding and tuning of parameters (e.g. see [Tyrrell, 1993, Ch.9], [Aylett, 1995]) until the designer is satisfied that the formulas deliver utility estimates that are fair.

In fact, there may be a way that these utility values can come for free. Learning methods that automatically assign values to actions are common in the field of Reinforcement Learning (RL) [Kaelbling, 1993]. Reinforcement Learning propagates numeric rewards into behavior patterns. The rewards may be external value judgements, or just internally generated numbers. This paper compares eight different methods of further propagating these numbers to solve the action selection problem.

The low-level problem of pursuing a single goal can be solved by straightforward RL, which assumes such a single goal. For the high-level problem of choice between conflicting goals we try various methods exploiting the low-level RL numbers.

1.1 Multi-module Reinforcement Learning

In general, Reinforcement Learning work has concentrated on problems with a single goal. For complex problems, that need to be broken into subproblems, most of the work either designs the decomposition by hand [Moore, 1990], or deals with problems where the sub-tasks have termination conditions and combine sequentially to solve the main problem [Singh, 1992, Tham and Prager, 1994].

The action selection problem essentially concerns subtasks acting in parallel, and interrupting each other rather than running to completion. Typically, each subtask can only ever be *partially* satisfied [Maes, 1989]. Lin has devised a form of multi-module RL suitable for such problems [Lin, 1992], and this will be the second method tested below.

2 The House Robot problem

We will demonstrate six of the methods in use in the hypothetical world of a 'house robot' (the final two are merely described). The house robot is given a range of multiple parallel and conflicting goals and must partially satisfy them all as best as it can. It doubles as a mobile security camera, mobile smoke alarm and occasional vacuum cleaner.

The artificial gridworld of Figure 1 was not constructed to be a *simulation* of a robot environment but

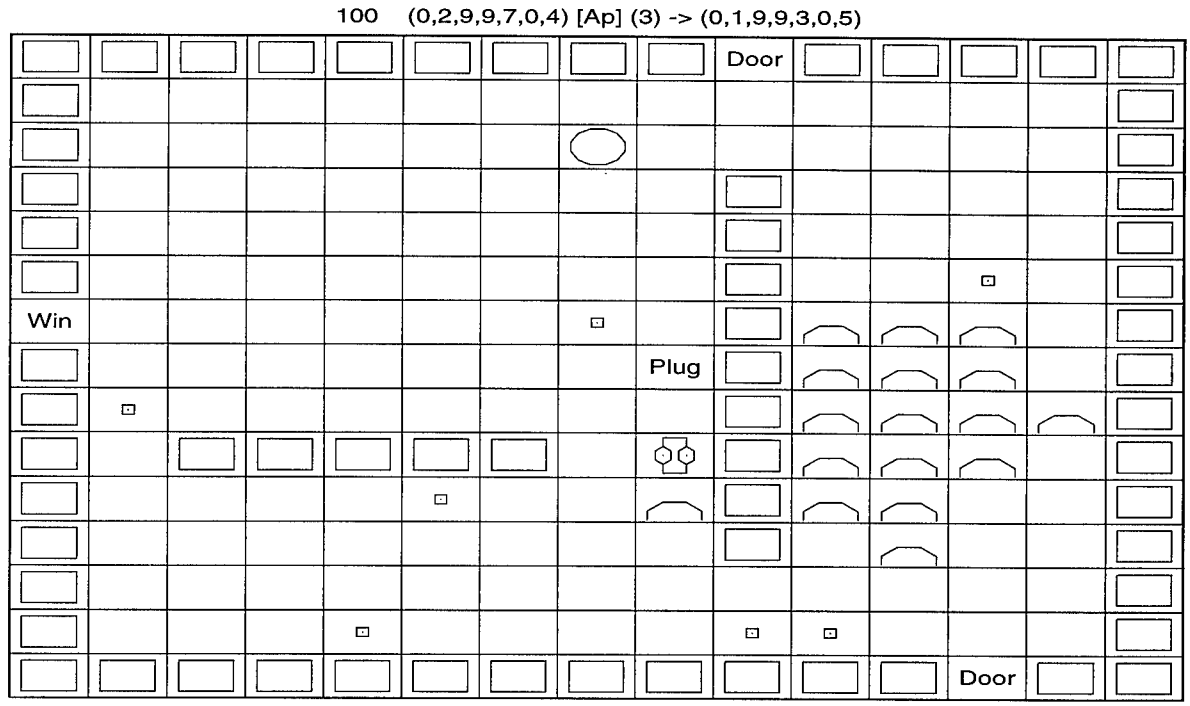


Figure 1: The House Robot's world. Here, the building is on fire, dirt is scattered everywhere, and an unidentified human has just come in the top door.

just to be a dynamic world in which to explore multiple conflicting goals. The positions of entrances and internal walls are randomised on each run. Humans are constantly making random crossings from one entrance to another. The robot or animat should follow strangers, and stay out of the way of family. It must go up close first to identify the human as family or stranger. Dirt trails after all humans. The animat picks up dirt and must occasionally return to some base to re-charge and empty its bag. Fire starts at random and then grows by a random walk. The animat puts out the fire on a square by moving onto it. Each time step, the animat senses state $x = (d, i, p, w, h, c, f, w_f)$, where:

- d is the direction (but not distance) of the nearest visible dirt, and takes values 0-7 (the primary and secondary compass directions), 8 (when dirt is on the same square) and 9 (no dirt visible within a small radius).
- i is whether the vacuum bag is full and needs emptying, and takes values 0 and 1.
- p (0-9) is the direction of the plug.
- w (0-9) is the direction of the nearest visible wall.
- h (0-9) is the direction of the nearest visible human.
- c is the classification of the human, taking values 0 (no current classification), 1 (known member of family) and 2 (stranger).
- f (0-9) is the direction of the nearest visible smoke. Smoke is the only thing that it can detect through walls - otherwise if something is blocked by a wall it is not visible.
- w_f is whether the smoke is being detected through a wall, and takes values 0 and 1.

The animat takes actions a, which take values 0-7 (move in that direction) and 8 (stay still).

Given this sensory information, the animat needs to develop a purely reactive strategy to put out fire, clean up dirt, watch strangers, and regularly return to base to re-charge. When we specify precisely (see next section) what we want, we find that the optimum is not any strict hierarchy of goals. Rather some interleaving of goals is necessary, with different goals partially satisfied on the way to solving other goals. Such goal-interleaving programs are difficult to write and make good candidates for learning.

3 Q-learning

The first method we apply is a single monolithic agent learning from rewards. Watkins [Watkins, 1989] introduced the method of reinforcement learning called *Q-learning*. Each discrete time step, the agent observes state x, takes action a, observes new state y, and receives immediate reward r. The agent is interested not just in immediate rewards, but in the *total discounted*

reward. In this measure, rewards received n steps into the future are worth less than rewards received now, by a factor of γ^n where $0 \leq \gamma < 1$:

$$R = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots$$

The strategy that the Q-learning agent adopts is to build up *Quality-values* (Q-values) for each pair (x, a). In 1-step Q-learning, after each experience, we update:

$$Q(x, a) \longrightarrow (r + \gamma \max_{b \in A} Q(y, b))$$

where $Q(x, a) \longrightarrow d$ means that we adjust the estimate $Q(x, a)$ in the direction of d. For example, if we store each $Q(x, a)$ explicitly, we may update: $Q(x, a) := (1 - \alpha)Q(x, a) + \alpha d$, where α is the learning rate. Or if the function is being approximated by a neural network, we backpropagate the error $Q(x, a) - d$.

All Q-values start randomised. Given some restrictions, [Watkins and Dayan, 1992] proved that for lookup tables Q-learning converges to a unique set of values $Q^*(x, a)$ which define a stationary deterministic optimal policy, namely to always take the action with the highest Q^*-value.

3.1 A global reward function

Reinforcement learning is attractive because it propagates rewards into behavior, and presumably reward functions (value judgements) are easier to design than behavior itself. Even so, designing the global reward function here is not easy (see [Humphrys, 1996] for an example of accidentally designing one in which the optimum solution was to jump in and out of the plug nonstop). Later we will ask if we can avoid having to design this explicitly.

```
reward for single step from x to y

 points := 0
 once-off type scores:
  if (got in way of family) subtract 1 point
  if (picked up dirt)            add 1 point
  if (put out fire)              add 5 points
 continuous-type scores:
  if (arrived at plug)             add 0.1 points
  if (stranger exists unseen) subtract 0.1 points
  if (fire exists)            subtract 0.1 points
  if (fire is large)          subtract 0.5 points
 return points
```

3.2 Neural network implementation

The number of possible states x is 1.2 million, and with 9 possible actions we have a state-action space of size 10.8 million. To hold each $Q(x, a)$ explicitly as a floating point number, assuming 4 byte floats, would therefore require 40 M of memory, which on my machine anyway was impractical. So instead of using lookup tables, we need to use some sort of generalization - here, multi-layer neural networks.

Following [Lin, 1992], because we have a small finite number of actions we can reduce interference by breaking the state-action space up into one network per action. We have 9 separate nets acting as function approximators. Each takes a vector input x and producing a floating point output $Q_a(x)$

Furthermore, as in [Rummery and Niranjan, 1994], we note here that each element of the input vector x takes only a small number of discrete values. So instead say of one input unit for (d) taking values 0-9, we can have 10 input units taking values 0 or 1 (a single unit will be set to 1, all the others set to 0). This makes it much easier for the network to identify and separate the inputs. Employing this strategy, we represent all possible inputs x in 57 input units which are all binary 0 or 1. Also like [Rummery and Niranjan, 1994], we found that a small number of hidden units (10 here) gave the best performance.

As [Tesauro, 1992] notes, learning here is not like ordinary supervised learning where we learn from (Input,Output) exemplars. Here we're not presenting $(x, Q^*(x))$ exemplars to the network but instead we are learning from *estimates* of Q^*. We need to repeat the updates as the estimate improves. Our strategy roughly follows [Lin, 1992]. We do 100 trials. In each trial we interact with the world 1400 times, remember our experiences, and then *replay* the experiences 30 times, each time factoring in more accurate Q estimates. Like Lin, we use *backward* replay as more effective (update $Q(y)$ before updating the $Q(x)$ that led to it). Throughout the experiments in this paper, we use $\gamma = 0.6$, and all lookup tables and neural networks (and hence Q-value estimates) start randomised.

Adjusting the amount of replay, and the architecture of the networks, the most successful monolithic Q-learner, tested over 20000 steps (involving 30 different randomised houses) scored an average of 6.285 points per 100 steps. This is not an optimal policy - writing a strict hierarchical program to solve the problem, with attention devoted to humans only when there was no fire, and attention devoted to dirt only when there was no fire or humans, could achieve a score of 8.612. Q-learning did not find an optimal policy because we are not using lookup tables, and do not have time anyway to experience each full state.

Clearly, it is difficult to learn such a complex single mapping. We will now look at ways in which the learning problem may be broken up.

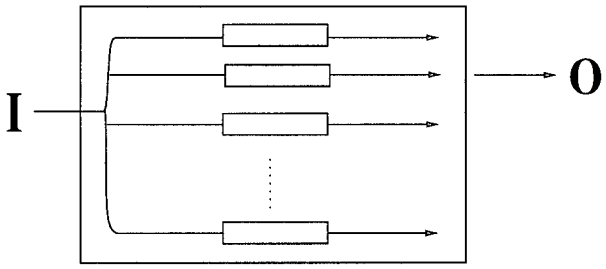


Figure 2: Competition among peer agents in a horizontal architecture. Each agent suggests an action, but only one action is executed. Which agent is obeyed changes dynamically.

4 Hierarchical Q-learning

[Lin, 1993] suggests breaking up a complex problem into sub-problems, having a collection of Q-learning agents [1] $A_1, \ldots, A_n$ learn the sub-problems, and then a single controller Q-learning agent which learns $Q(x, i)$, where i is which agent to choose in state x. This is clearly an easier function to learn than $Q(x, a)$, since the sub-agents have already learnt sensible actions. When the animat observes state x, each agent A_i suggests an action a_i. The switch chooses a winner k and executes a_k.

Lin concentrates on problems where subtasks combine to solve a global task, but one may equally apply the architecture to problems where the sub-agents simply compete and interfere with each other, that is, to classic action selection problems. In this decentralised approach (Figure 2), each behavior will drive the body on its own if allowed, but must send its actions to a switch which will either obey or refuse it. In hierarchical Q-learning, the switch is complex and what is sent is simple (a constant action a_i). Later, we will make the switch simple, and what is sent more complex.

We set the following 5 agents to build up personal $Q_i(x, a)$ values, where the x they sense ranges over a subspace small enough to use lookup tables, and they learn by reference to personal reward functions r_i. The switch then learns $Q(x, i)$ from the global reward function. The switch still sees the full state x, and needs to be implemented as 5 neural networks.

A_d	senses: (d,i)	
	reward: if (picked up dirt)	1 else 0
A_p	senses: (p)	
	reward: if (arrived at plug)	1 else 0
A_s	senses: (h,c)	
	reward: if (ID=stranger and visible)	1 else 0
A_m	senses: (h,c)	
	reward: if (ID=family and here)	0 else 1
A_f	senses: (f,w_f)	
	reward: if (put out fire)	1 else 0

[1] I still use the word *agent*, even though we now have multiple agents inside the same body. This is similar to the use of the word in [Minsky, 1986].

Here the agents share the same suite of actions $a = 0$-8, but in general we may be interested in breaking up the action space also. We try to keep the number of agents low, since a large number of agents will require a large (x, i) statespace. Note that having all rewards set to 1 doesn't reduce our options - if we replace 1 above by 0.5, or any number > 0, the agent still learns the same pattern of behavior. It still sends the same preferred action a_i to the switch. This does not hold true with a 3-reward (or more) reward function, where relative differences between rewards matters.

With the same replay strategy as before, and the same number of test runs, the hierarchical Q-learning system scored 13.641 points per 100 steps, a considerable improvement on the single Q-learner.

5 W-learning (Minimize the Worst Unhappiness)

Looking at exactly what $Q(x, i)$ are learnt in the previous method, the switch learns things like - if dirt is visible A_d wins because it expects to make a good contribution to the global reward - otherwise if the plug is visible A_p wins because it expects to make a (smaller) contribution. But the agents could have told us this themselves, with reference only to their personal rewards. In other words, the agents could have organised themselves like this in the absence of a global reward.

W-learning [Humphrys, 1995] is an attempt to define a sensible way for the agents to organise themselves in the absence of a global reward. The agents learn by compromising, and the agent that wins is the agent that would suffer the most if it did not win. Given a state x, the agents suggest their actions with strengths or *Weights* $W_i(x)$. The switch in Figure 2 becomes a simple gate letting through the highest one. When the animat observes state x, the switch finds the winner k such that:

$$W_k(x) = \max_{i \in 1,\ldots,n} W_i(x)$$

and executes a_k. We call A_k the *leader* in the competition for state x at the moment, or the *owner* of x at the moment. The agents then *modify* their $W_i(x)$ values based on whether they were obeyed (and what happened if they weren't), so that next time round there may be a different winner. Schemes using such 'importance' values are common in multi-behavior models (e.g. see the 'utility' functions in [Aylett, 1995]), but are normally hand-designed.

W-learning builds up the difference between predicted reward $\mathcal{P}$ (what is predicted if the agent is obeyed) and actual reward $\mathcal{A}$ (what actually happened). Consider Q-learning as the process:

$$\mathcal{P} \longrightarrow \mathcal{A}$$

where we are *learning* $\mathcal{P}$, and $\mathcal{A}$ is caused by the execution of our action. Then W-learning is:

$$W \longrightarrow (\mathcal{P} - \mathcal{A})$$

where $\mathcal{P}$ is *already* learnt, and $\mathcal{A}$ is caused by the execution of *another* agent's action. $(\mathcal{P} - \mathcal{A})$ is the 'error' or loss that the other agent is causing for this one by being obeyed in its place. When we see a difference term between predicted and actual, we expect that this 'error term' will go to zero, but here it goes to a positive number. To be precise, when agent A_k is the winner and has its action executed, all agents *except* A_k update:

$$W_i(x) \longrightarrow (Q_i(x, a_i) - (r_i + \gamma \max_{b \in A} Q_i(y, b)))$$

where the reward r_i and next state y were caused by A_k rather than by the agent itself. W-learning does not assume that A_k's actions are meaningful to this agent - it does not assume we can look at $Q_i(x, a_k)$. But we can always observe what happened in terms of r_i and y. The error distribution we sample from will abruptly change if there is a change of leader in state x, but competition will eventually be resolved for this state when some agent, as a result of losses it has suffered in the past, builds up an unassailable W-value for x (see [Humphrys, 1995] for details).

Note that if we update the winner's W-value as well, we would be updating $W_k(x) \longrightarrow 0$, since if you are obeyed, your expected error is zero. So as soon as an agent gets into the lead, its W-value starts dropping until it loses the lead again. Competition is never resolved. This is why the leader does nothing - it's up to the others to catch up with it. If they can't, we have a resolved competition. Not updating $W_k(x)$ though, means that if it gets arbitrarily set to an unfairly high value, it will never be challenged. So we initialise all $W_i(x)$ to zero or random negative values.

5.1 *Making agents weaker or stronger*

There being no global (x, i) statespace to worry about, we can expand the number of agents. To the previous five, we add three more agents. A_c should head for the centre of an open area while A_w should engage in wall-following. We can add more agents than probably needed - if they're not useful they just won't win any W-competitions and won't be expressed. Rewards are in the range $0 < r \leq 1$. Both the $Q_i(x, a)$ values and $W_i(x)$ values refer to x in the little subspaces, for which lookup tables can be used.

A_d	senses: (d,i)	
	reward: if (picked up dirt)	r_d else 0
A_p	senses: (p)	
	reward: if (arrived at plug)	r_p else 0
A_c	senses: (w)	
	reward: if (lost sight of wall)	r_c else 0
A_w	senses: (w)	
	reward: if (wall same dir as last time)	r_w else 0
A_u	senses: (h,c)	
	reward: if (made ID)	r_u else 0
A_s	senses: (h,c)	
	reward: if (ID=stranger and visible)	r_s else 0
A_m	senses: (h,c)	
	reward: if (ID=family and here)	0 else r_m
A_f	senses: (f,w_f)	
	reward: if (put out fire)	r_f else 0

Unlike in the previous method, here the values of the rewards r_i *do* matter. An agent with rewards 1 and 0 will end up with higher W-values than an equivalent agent with rewards 0.1 and 0 and the same logic, since its absolute $(\mathcal{P} - \mathcal{A})$ differences will be greater. It will win a greater area of state-space in competitions. We make agents *stronger* (more influential in the collection) by increasing the differences between their rewards. Adaptive collections are likely to involve well-chosen combinations of weak and strong agents. The best combination found above was:

$r_d = 0.93$
$r_p = 0.01$
$r_c = 0.41$
$r_w = 0.01$
$r_u = 0.54$
$r_s = 0.60$
$r_m = 0.67$
$r_f = 0.67$

This was discovered by running a genetic algorithm search on combinations of r_i's. Given a particular combination, the agents learn their behaviors by Q-learning, and then organise their action selection by W-learning, without reference to a global reward. Obviously, if the global reward function still *defines* what we are looking for, we still need to use it - as the fitness function to guide our search. But it no longer need be available explicitly to the agents. It need only be used to test them. Hence the fitness function could be just *implicit* in the environment, as in the best Artificial Life research [Ray, 1991].

This combination scored 13.446, slightly less than we got with Hierarchical Q-learning, but achieved with a reduction in memory requirements from 9.6 million to 1600. For analysis of how the agents interact to achieve the task see [Humphrys, 1996]. Note that all GA searches in these experiments were population size ≤ 60, initially randomised, evolving for ≤ 30 generations.

6 W=Q (Maximize the Best Happiness)

The first response to W-learning is to ask if we need such an elaborate value of W. Why not simply have actions promoted with their Q-values. The agent promotes its action with the same strength no matter what (if any) its competition:

$$W_i(x) = Q_i(x, a_i)$$

and we just search for adaptive combinations as before. This works very well, and the following collection achieved a score of 15.313. Further, the memory requirements are even less, since no W-values at all are kept.

$r_d = 0.93$
$r_p = 0.41$
$r_c = 0.41$
$r_w = 0.08$
$r_u = 0.80$
$r_s = 0.14$
$r_m = 0.08$
$r_f = 1.00$

But we are not finished with W-learning yet. It seems on paper that W=Q should not perform so well, since it maximises the rewards of only one agent, while W-learning makes some attempt to maximise their collective rewards (which is roughly what the global reward is). Consider the following scenario, where there are two actions (1) and (2). The agents' preferred actions are highlighted:

```
a         (1)     (2)
Q1(x,a)   [1.1]   1
Q2(x,a)   0       [0.9]
```

If we use W=Q, then agent A_1 wins (since $1.1 > 0.9$), action (1) is executed, A_1 gets reward 1.1, and A_2 gets 0. If we use $W = (\mathcal{P} - \mathcal{A})$, then A_2 wins (since it would suffer 0.9 if it didn't), (2) is executed, A_1 gets 1, and A_2 gets 0.9. If the global reward / evolutionary fitness is roughly a combination of the agents' rewards, then $W = (\mathcal{P} - \mathcal{A})$ is a better strategy. Note that this is the familiar ethology problem of *opportunism* - can A_2 force A_1 into a small diversion from its plans to pick up along the way a goal of its own?

So why did W-learning not perform better? The answer seems to be that the house robot environment does not contain problems of the nature above. It contains situations where A_2 wants to slightly divert A_1 alright, but only in situations where A_2 itself doesn't mind being diverted - the 0 above becomes a 0.8. This is because all behaviors here are essentially of the form 'if some feature is in some direction, then move in some direction' with rewards for arriving at the feature or losing sight of it. So if $Q_1(x, 1) = 1.1$ is similar to $Q_1(x, 2) = 1$, it is because actions (1) and (2) are movements in roughly the same direction, in which case $Q_2(x, 1)$ and $Q_2(x, 2)$ will end up similar.

7 W-learning with full space

W-learning performed similarly to Hierarchical Q-learning, but with far less memory requirements. But note that using subspaces for $W_i(x)$ results in a loss of accuracy.

Consider the competition between the dirt-seeker A_d and the smoke-seeker A_f. For simplicity, let the global state be $x = (d, f)$. A_d sees only states (d), and A_f sees only (f). When the full state is $x = (d, 5)$, A_f simply sees all these as state (5), that is, smoke is in direction 5. Sometimes A_d opposes it, and sometimes, for no apparent reason, it doesn't. But $W_f(5)$ *averages* all these together into one variable. It is a crude form of competition, since A_f must present the same W-value in many different situations where its competition will want to do quite different things. The agents might be better able to exploit their opportunities if they could tell the real states apart and present different W-values.

If we are to make the x in the $W_i(x)$ refer to the full state, then each agent needs a single neural network to implement the function. Recall that the W-learning strategy for lookup tables is to start with W random negative, and have the leading $W_k(x)$ unchanged, waiting to be overtaken. This will not work with neural networks. First because trying to initialise W to random negative is pointless since the network's values will make large jumps up and down in the early stages when its weights are untuned. Second because even if we do not update it, $W_k(x)$ will still change as the other $W_k(y)$ change. And if the net doesn't see $W_k(x) \longrightarrow d$ for a while, it will forget it.

We could think of various methods to try to repeatedly clamp $W_k(x)$, but it seems all would need extra memory to remember what value it should be clamped to. The simplest approach is probably: Start with W random. Do one run of 30000 steps with *random* winners so that everyone experiences what it's like to lose, and remembers these experiences. Then they all replay their experiences 10 times to learn from them properly. We use a similar network architecture as before. The best combination of agents found, scoring 14.871, was:

$r_d = 0.67$
$r_p = 0.01$
$r_c = 0.80$
$r_w = 0.08$
$r_u = 0.14$
$r_s = 0.60$
$r_m = 0.21$
$r_f = 1.00$

8 Negotiated W-learning

If other agents' actions are meaningless to it, all an agent can do is observe what r and y they generate, as W-learning does. However, if other agents' actions mean something to the agent, it already has built up an estimate of the expected reward in the value $Q_i(x, a_k)$. So rather than learning a W-value from samples, it can assign it directly if the successful action a_k is communicated to it. We can do this in the house robot problem,

since all agents share the same suite of actions ('move' 0-8).

In Negotiated W-learning, the animat observes a state x, and then its agents engage in repeated rounds of negotiation before resolving competition and producing a winning action a_k. It is obviously to be preferred that the length of this competition will be very short. The 'instant' competition operates as follows. Each time step:

```
observe state x

start with leader k := random agent and Wk := 0
loop:
 for all agents i other than k
   Wi := Qi(x,ai) - Qi(x,ak)
 if highest Wi > Wk, new leader and goto loop

(loop has terminated with winner k)
execute ak
```

This algorithm discovers explicitly in one timestep what W-learning only learns over time. It also gives us the high accuracy of telling states apart, as in W-learning with full statespace, yet without any memory requirements at all for W. In fact, note that 'Negotiated W-learning' is actually not learning at all since nothing permanent is learnt. The best combination found, scoring 18.212, was:

$r_d = 0.87$
$r_p = 0.01$
$r_c = 0.54$
$r_w = 0.01$
$r_u = 1.00$
$r_s = 0.08$
$r_m = 0.74$
$r_f = 0.34$

It is easily seen [Humphrys, 1995] that the competition length will be bounded by 1 and $n + 1$ (remember here $n = 8$). With the combination above, the competition lengths seen over a run of 40000 steps (actually, for technical reasons, 39944 steps) were:

1	234	0.6%
2	27164	68.0%
3	11978	30.0%
4	558	1.4%
5	10	0.025%
6	0	
7	0	
8	0	
9	0	

This gives a (reasonably reactive) average competition length of 2.3 (Figure 3).

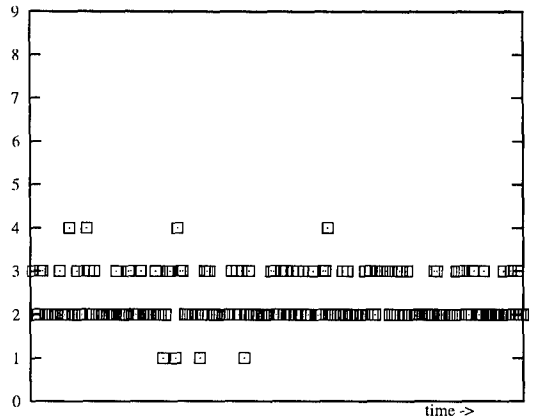


Figure 3: The 'reactiveness' of Negotiated W-learning. This is a typical snapshot of 200 steps of the best solution found, showing how long it took to resolve competition at each timestep. The theoretical maximum competition length is 9 (the number of agents plus 1).

9 Maximize Collective Happiness

For completeness we describe two final methods, though they have not yet been tested empirically. First, if the global reward is roughly the sum of the agents' rewards, maybe we should *explicitly* maximize collective rewards. If the agents share the same suite of actions, we can calculate:

$$\max_{a \in A} \left[\sum_{i=1}^{n} Q_i(x, a) \right]$$

Note that this may produce *compromise* actions. The executed action may be an action that none of the agents would have suggested.

10 Minimize Collective Unhappiness

Obviously, the final missing method is to *minimize collective unhappiness*. In this method, each agent builds up a value $W_i(x)$ which is the sum of the suffering it causes *all the other agents* when it is being obeyed. We look for the smallest $W_i(x)$. Like W-learning, agents do not need to share common actions. Rather, they observe r_i and y, and build up their deficits over time. We start with all $W_i(x)$ zero or random negative. Each time step:

```
observe state x
find Wk(x) = lowest Wi(x)
execute ak
all agents i other than k add to Wk(x)
 (so that it might not win next time round)
```

Again like W-learning, if agents *do* share common actions, we can resolve this immediately rather than waiting for W-values to build up over time.

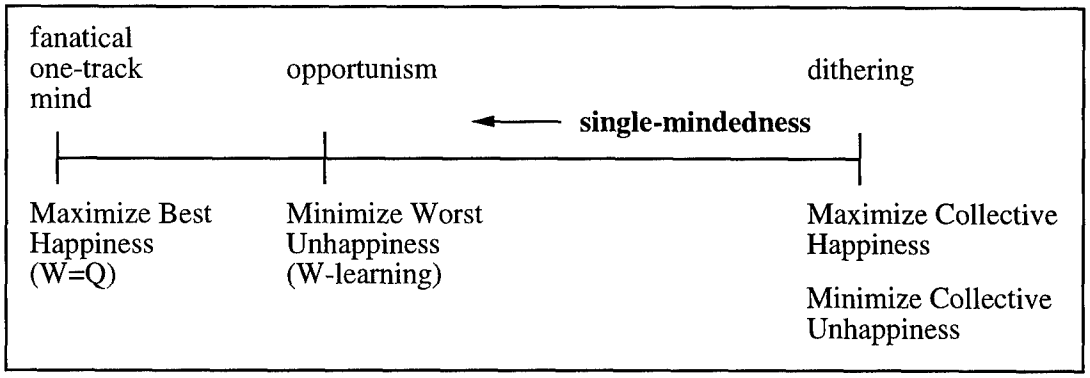


Figure 4: The 'single-mindedness' of the methods that organise action selection without reference to a global reward.

	Memory requirements n no.agents x subspace X full space	No. updates per timestep (per agent) when learning	No. updates per timestep when exploiting
Q-learning	1.Xa	1	0
Hierarchical Q-learning	n.xa + 1.Xn	2	0
W=Q	n.xa	1	1
Max Collective Happ.	n.xa	1	n/applicable
W-learning (subspaces)	n.xa + n.x	2	0
W-learning (full space)	n.xa + n.X	2	0
Negotiated W-learning	n.xa	1	1 to n+1
Min Collective Unhapp.	n.xa + n.x	n	0

General comparisons between the methods.

	Memory requirements	No. updates per timestep when learning	No. updates per timestep when exploiting	Best solution found
Hand-coded program (strict hierarchical)	n/applicable	n/applicable	n/applicable	8.612
Q-learning	10800000	1	0	6.285
Hierarchical Q-learning	9601440	2	0	13.641
W=Q	1440	1	1	15.313
Max Collective Happ.	1440	1	n/applicable	n/tested
W-learning (subspaces)	1600	2	0	13.446
W-learning (full space)	9601440	2	0	14.871
Negotiated W-learning	1440	1	average 2.3	18.212
Min Collective Unhapp.	1600	8	0	n/tested

Comparisons between the methods as applied in the house robot problem.

10.1 Expected performance of the collective methods

The two collective methods will generate the same sort of behavior - keeping the majority of agents happy at the expense perhaps of a small minority. Collective approaches are probably a bad idea if there are a *large* number of agents. The animat will choose safe options, and no one agent will be able to persuade it to take risks. Even if one agent is facing a *non-recoverable* state (where if it is not obeyed now, it cannot ever recover and reach its goal in the future), it may still not be able to overcome the opinion of the majority.

One can easily set up situations in which trying to keep the majority of agents happy will lead to problems with *dithering*, in which no goal is pursued to its logical end. W=Q goes to the other extreme in having only one agent in charge, and perhaps suffers because it does not allow *opportunism*. W-learning may be a good balance between the two, allowing opportunism without the worst of dithering. One agent is generally in charge, but will be corrected by the other agents whenever it offends them too much.

[Maes, 1989] lists desirable criteria for action selection schemes, and in it we see this tension between wanting actions that contribute to several goals at once and yet wanting to stick at goals until their conclusion. We can represent this in a diagram of 'single-mindedness' (Figure 4).

11 Conclusion

In conclusion, it seems that selfishness on the part of agents is justified - compare (see tables) the stronger performance of the various W methods. Agents can spot better than global functions opportunities to pick up rewards, so letting agents be selfish is more likely to maximise their combined rewards.

Whether it is better to build up differences ($\mathcal{P} - \mathcal{A}$) (the three W-learning methods) or just use W=Q is perhaps still an open question. Although the best one was one of the W-learning methods, W=Q was second best. Perhaps, as discussed in that section, opportunism isn't important enough in this particular problem world to separate the two approaches.

Finally, it is remarkable how difficult it is to hand-code the goal interleaving necessary to perform well in this world. The solutions generated by learning are far superior, though difficult to visualise or translate into a concise set of instructions.

12 Further work

As discussed, to further compare these methods, they could be tested in an environment in which the rewards of opportunism (and the costs of not being opportunistic) are greater. The collective methods should be tested in situations where some agents have non-recoverable states.

12.1 Competition in Single Goal problems

Note that with the W methods, agents don't actually have to be solving different goals. This may be a novel approach to classic single-goal problems. To solve a problem, we put together a large number of agents, all with different reward functions (and perhaps even different senses), all roughly trying to solve the same thing, and let them struggle to solve it. If there are multiple unexpressed behaviors, and lots of overlap, this may be a small price to pay if there is a robust solution. We could be similarly profligate with the number of agents in Hierarchical Q-learning too, but would pay the price of a large (x, i) statespace.

12.2 Ecosystem Minds

This work was partly inspired by Edelman's vision of a "rainforest" mind, with life and death, growth and decay, in a dynamic ecosystem inside the head [Edelman, 1989, Edelman, 1992]. The idea is appealing, but Edelman presents no explicit algorithm to show how it could be implemented. [2] Here, with W-learning, we have the basis for a full living and dying ecosystem inside the animat, where what comes into existence, struggles, and perhaps dies is not mere connections (as Edelman may in fact mean) but entire autonomous goals.

In both W=Q and Negotiated W-learning, nothing needs to be re-learnt when new agents arrive or old ones leave, but the dynamics of the system will immediately change in both. It seems Negotiated W-learning would be more robust since agents may generate higher W-values in the face of new competition, while in W=Q they are stuck with fixed W-values no matter what the competition. We could even imagine collections which are difficult for new agents to invade - where W-values are currently low but would all rise, as if in defence, on arrival of the invader. Such would be not only an ecosystem mind, but a stable, elderly ecosystem mind, set in its ways.

References

[Aylett, 1995] Aylett, Ruth (1995), Multi-Agent Planning: Modelling Execution Agents, in Sam Steel, ed., *Papers of the 14th Workshop of the UK Planning and Scheduling Special Interest Group.*

[2] In fact, presenting arguments only about classical AI, he draws the familiar conclusion that no computer algorithm could implement his ideas. No mention is made of self-modifying, reinforcement-learning agents embedded in a world, to which his criticisms do not apply.

[Blumberg, 1994] Blumberg, Bruce (1994), Action-Selection in Hamsterdam: Lessons from Ethology, in Dave Cliff et al., eds., *Proceedings of the Third International Conference on Simulation of Adaptive Behavior (SAB-94).*

[Brooks, 1991] Brooks, Rodney A. (1991), Intelligence without Representation, *Artificial Intelligence* 47:139-160.

[Edelman, 1989] Edelman, Gerald M. (1989), *The Remembered Present: A Biological Theory of Consciousness*, Basic Books.

[Edelman, 1992] Edelman, Gerald M. (1992), *Bright Air, Brilliant Fire: On the Matter of the Mind*, Basic Books.

[Humphrys, 1995] Humphrys, Mark (1995), *W-learning: Competition among selfish Q-learners*, technical report no.362, University of Cambridge, Computer Laboratory. `http://www.cl.cam.ac.uk/users/mh10006/publications.html`

[Humphrys, 1996] Humphrys, Mark (1996), Action selection in a hypothetical house robot: Using those RL numbers, in Peter G.Anderson and Kevin Warwick, eds., *Proceedings of the First International ICSC Symposia on Intelligent Industrial Automation (IIA-96) and Soft Computing (SOCO-96).*

[Kaelbling, 1993] Kaelbling, Leslie Pack (1993), *Learning in Embedded Systems*, The MIT Press/Bradford Books.

[Lin, 1992] Lin, Long-Ji (1992), Self-Improving Reactive Agents Based On Reinforcement Learning, Planning and Teaching, *Machine Learning* 8:293-321.

[Lin, 1993] Lin, Long-Ji (1993), Scaling up Reinforcement Learning for robot control, *Proceedings of the Tenth International Conference on Machine Learning.*

[Maes, 1989] Maes, Pattie (1989), How To Do the Right Thing, *Connection Science* vol.1, no.3.

[Minsky, 1986] Minsky, Marvin (1986), *The Society of Mind*, Simon and Schuster, New York.

[Moore, 1990] Moore, Andrew W. (1990), *Efficient Memory-based Learning for Robot Control*, PhD thesis, University of Cambridge, Computer Laboratory.

[Ray, 1991] Ray, Thomas S. (1991), An Approach to the Synthesis of Life, in Christopher G.Langton et al., eds., *Artificial Life II.*

[Rummery and Niranjan, 1994] Rummery, Gavin and Niranjan, Mahesan (1994), *On-line Q-learning using Connectionist systems*, technical report no.166, University of Cambridge, Engineering Department.

[Sahota, 1994] Sahota, Michael K. (1994), Action Selection for Robots in Dynamic Environments through Inter-behaviour Bidding, in Dave Cliff et al., eds., *Proceedings of the Third International Conference on Simulation of Adaptive Behavior (SAB-94).*

[Singh, 1992] Singh, Satinder P. (1992), Transfer of Learning by Composing Solutions of Elemental Sequential Tasks, *Machine Learning* 8:323-339.

[Tesauro, 1992] Tesauro, Gerald (1992), Practical Issues in Temporal Difference Learning, *Machine Learning* 8:257-277.

[Tham and Prager, 1994] Tham, Chen K. and Prager, Richard W. (1994), A modular Q-learning architecture for manipulator task decomposition, *Proceedings of the Eleventh International Conference on Machine Learning.*

[Tyrrell, 1993] Tyrrell, Toby (1993), *Computational Mechanisms for Action Selection*, PhD thesis, University of Edinburgh, Centre for Cognitive Science.

[Watkins, 1989] Watkins, Christopher J.C.H. (1989), *Learning from delayed rewards*, PhD thesis, University of Cambridge, Psychology Department.

[Watkins and Dayan, 1992] Watkins, Christopher J.C.H. .and Dayan, Peter (1992), Technical Note: Q-Learning, *Machine Learning* 8:279-292.

MPEG Movie demo



A series of MPEG Movies of W-learning operating in a simple 'antworld', where an animat must collect food while avoiding moving predators, can be viewed at: `http://www.cl.cam.ac.uk/users/mh10006/w.html`

Towards Adaptive Behaviour System Integration using Connectionist Infinite State Automata

Tom Ziemke
Neural Computing Group, Department of Computer Science, University of Sheffield, UK
&
Connectionist Research Group, Department of Computer Science, University of Skövde
Box 408, 54128 Skövde, Sweden
tom@ida.his.se

Abstract

A higher order recurrent connectionist architecture for adaptive control of autonomous robots is introduced in this paper. This architecture, inspired by Pollack's *Sequential Cascaded Network*, consists of two sub-networks: a *function network* for the coupling between sensory inputs and motor outputs, and a *context network*, which dynamically adapts the function network in order to allow a flexible mapping from percepts to actions.

The approach taken here is compared to dynamics and algorithmic approach to autonomous robot control, and it is argued that the above architecture allows an integration of (a) the complex structure and control of behaviour typical for the algorithmic approach, (b) the capacity to utilize systematically continuous state spaces, and (c) the self-organizing learning capacity of connectionist systems with a simple, but powerful mechanism for context-dependent adaptation of behaviour.

1 Introduction

Autonomous agents interacting with realistically complex and dynamic environments are usually facing a (possibly large) number of different and varying tasks, situations and requirements whose integration often is far from trivial. In the area of *behaviour-oriented AI* (cf. Steels, 1994) the overall control of such systems therefore usually is considered to be best realized by (or emerge from) a number of behaviour-systems, each of which is responsible for establishing a particular behaviour (i.e. a regularity in the interaction dynamics between agent and environment).

The concept of true autonomy is however not limited to automaticity, i.e. the capacity to interact with an environment, it further includes the ability to (continually) learn from this interaction, i.e. *the individual agent's own capacity (at least partly) to form and adapt its principles of behaviour* (Steels, 1995).

The next section briefly discusses different approaches to the synthesis of adaptive behaviour in robots, and in particular their suitability for providing the latter capacity. Section 3 introduces a recurrent connectionist architecture for self-adapting networks, aiming at increasing a robotic agent's degree of adaptivity and autonomy. Furthermore a number of experiments with this architecture are presented and evaluated. Finally, section 4 summarizes the work presented here, and presents a number of conclusions and directions for future work.

2 Approaches in Behaviour-Oriented AI

Different approaches to the design of behaviour systems and underlying behaviour programs roughly fall in the three groups (cf. Steels, 1994) briefly discussed in the following sections. A fourth approach to the synthesis of adaptive behaviour is that of evolutionary learning, which however typically does not address the problem of behaviour adaptation in *individual agents*, and therefore is excluded from the discussion in this paper.

2.1 Algorithmic Approach - Finite State Automata

Algorithmic approaches are typically based on a behaviour-oriented decomposition of the overall control task (as for example into wandering, obstacle avoidance, etc.), and usually result in a corresponding modularisation of the internal control mechanism, usually in some form of finite-state automaton (FSA) (e.g. Brooks' *subsumption architecture* (1986) and its descendants). This class of systems suffers from the following problems (Steels, 1994): a) in a concrete agent the number of states and corresponding subsystems quickly increases, and so does their complexity, and b) the inherent commitment to FSA as underlying mechanism results in a finite number of discrete states which makes it difficult to achieve smooth, continuous behaviour at the level of the overall system.

Furthermore, since the above mentioned modularisation is not acquired through self-learning but usually carried out top-down and a priori by the system's designer, the controlled system a) has not at all formed 'its principles of behaviour' itself, and b) it is doubtful how it could ever substantially adapt or extend them itself, since that would

require to dynamically restructure or extend the module hierarchy on-line which has not been shown to be feasible yet. Hence, the controlled system might very well be automatic, it can however not be called autonomous in the above sense, since it is not in control of the principles of its own behaviour.

2.2 Dynamics Approaches

Dynamics approaches (e.g. Steels, 1993) are based on the notion of behaviour systems as continuous dynamical systems instead of discrete computational systems as in the algorithmic approach. A dynamics architecture typically consists of a number of processes, each establishing a continuous relationship between a set of quantities (e.g. sensory inputs and action parameters), describable by a set of differential equations. Typically processes/behaviour systems are active all the time and 'cooperate' through combination of their effects at the level of actions (i.e. if for example two independent subsystems suggest turns of 20 degrees to the left and 20 degrees to the right the result could be not to turn at all). This reduction of the overall control to a simple combination of the effects of individual behaviour systems does in general lead to smoother behaviour. It does however also pose a problem since it makes a more complex integration or structural coupling of behaviours (e.g. one behaviour overriding another) very difficult. Hence, more complex control situations require the extension of the 'pure' framework of differential equations through introduction of 'motivational variables' (e.g. 'hunger') that causally influence behaviour systems and have a dynamics of their own (cf. Steels, 1994).

Learning and behaviour adaptation in this framework are typically easier to achieve than with the algorithmic approach, since to some extent they can be facilitated through adaptation of quantities/variables. However, when it comes to adaptation/learning that requires an increase/change of behavioural complexity or structure, this approach faces a problem that is analog to that of the algorithmic approach: it is unclear how an agents could carry out this adaption on its own, since that would for example require to add new differential equations or motivational variables.

2.3 Connectionist Approaches

Connectionist approaches are usually based on a (more or less) direct coupling between a system's sensory inputs and action parameter outputs through artificial neural networks (ANNs). Especially recurrent networks, a special case of dynamical systems, are often used for this purpose (e.g. Meeden, 1996, Tani, 1996) as they allow an implicit representation of dynamical aspects in form of an internal state. A major advantage of this approach is that it incorporates a mechanism for learning behaviour programs which, often in combination with reinforcement learning techniques, allows a bottom-up development of integrated control strategies (e.g. Meeden, 1996), which, in the simplest case, reduces the system designer's task to the choice of an appropriate architecture. Due to the fact that the functionality of an ANN is distributed over all connection weights and units in a network (rather than broken up and isolated in specific-purpose modules, equations or variables), an increase in behavioural complexity can typically be facilitated through a re-distribution of functionality (i.e. adaptation of weights), and does not require an increase in resources or structural complexity of the ANN itself.

The problem with this approach however is that, although ANNs might be very well suited for the realization of individual behaviour programs, a single (conventional, i.e. first order) ANN could hardly be sufficient to realize and integrate a number of complex and diverse behaviour systems as it would be required to build a complete autonomous agent (cf. Steels, 1994). More specifically the problem is that an ANN always realizes a certain function (or an iterated function system in the case of recurrent networks (cf. Kolen, 1994b)), i.e. a particular mapping of input (and internal state) to output. The weights in such a network, embodying this functional mapping, usually remain constant after a certain training phase, or, in continual or 'lifelong' learning approaches, are updated only stepwise and very gradually. That means that the complexity/diversity of behaviour that can be learned by a single (conventional) ANN is strongly limited by the degree to which a number of behaviour systems can be realized/integrated in a single functional mapping. This limits the adaptation of the controlled system/robot to a possibly rapidly changing environment.

3 Models & Analysis

3.1 The Architecture

The architecture proposed here aims at increasing the adaptivity of a connectionist network, and thereby the complexity and diversity of behaviour it can learn and integrate, by allowing it to dynamically adapt its own weights (and thereby its functional mapping) to its current context.

This architecture is mainly inspired by Pollack's *Sequential Cascaded Network* (SCN, 1991). The SCN (originally used for formal language recognition) consists of two subnetworks in a master-slave relationship: A one-layer *function (slave) network* maps the network's input vector (I) to state (S) and output (O) vector, and a linear one-layer *context (master) network*, which dynamically computes the input weights in the function network. Hence, the SCN can be described by the following two dynamic equations (cf. Kolen,1994a):

$$O(t) = g\left(\left(W \cdot \begin{bmatrix} S(t) \\ 1 \end{bmatrix}\right) \cdot \begin{bmatrix} I(t) \\ 1 \end{bmatrix}\right) \qquad S(t) = g\left(\left(V \cdot \begin{bmatrix} S(t-1) \\ 1 \end{bmatrix}\right) \cdot \begin{bmatrix} I(t) \\ 1 \end{bmatrix}\right)$$

Where g is the logistic function, W and V are three-dimensional weight matrices mapping S(t) to O(t) and S(t+1). The SCN is trained with a modified version of backpropagation.

The architecture proposed here (cf. figure 1) basically uses the same context network, but extends the function network to a form similar to Elman's *Simple Recurrent Network* (SRN, 1990), i.e. a three-layer network in which the hidden units are used as additional inputs in the next time step.

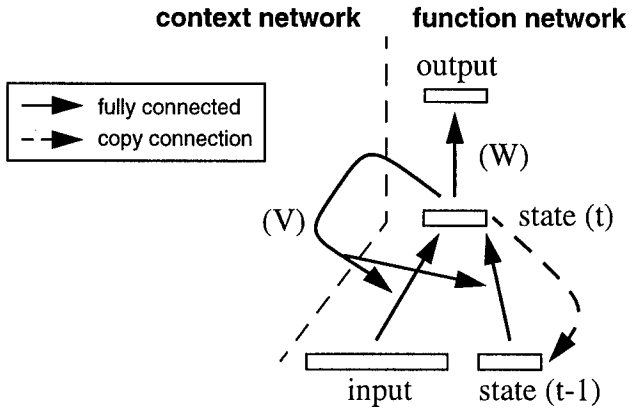


Figure 1: Self-Adapting Recurrent Network (SARN)

Hence, the proposed architecture can be described by dynamic equations as follows:

$$O(t) = g\left(W \cdot \begin{bmatrix} S(t) \\ 1 \end{bmatrix}\right) \qquad S(t) = g\left(\left(V \cdot \begin{bmatrix} S(t-1) \\ 1 \end{bmatrix}\right) \cdot \begin{bmatrix} I(t) \\ S(t-1) \end{bmatrix}\right)$$

Where g is the logistic function, W is a two-dimensional weight matrix mapping S(t) to O(t), and V is a three-dimensional weight matrix mapping S(t-1) to the next step's input weights.

The SARN is trained continually using a modified online form of the backpropagation algorithm (cf. section 3.3). First after each step the error at the output layer is backpropagated to update the state-to-output (W) and the input-to-state weights. Then the new input-to-state weights are used as target output for the context network to update the weights (V) between state layer and input-to-state weights.

It should be noted that figure 1 and the above equations describe a specific second order extension of the SRN as it has been used in the experiments documented in the following sections. For the overall discussion however this architecture should be considered a representative for a certain class of networks, namely ANNs that use recurrent connections to directly adapt their own connection weights. Possible variations within this class would be to add a hidden layer to the context network or to remove hidden layer or recurrent connections from the function network.

3.2 The Robot Simulator

The experiments described here were carried out with the Khepera Simulator (Michel, 1995), a simulated version of the real mobile Khepera robot (Mondada et al., 1993). The robot has two motors, each of which can move forward (in direction of sensors 2 and 3) and backward at different speeds (values between +10 and -10). Same speeds in opposite direction allow the robot to turn on the spot. The simulated mobile robot has eight infrared sensors (s. figure 2 for their position and direction) measuring both ambient light and distance from possible obstacles..

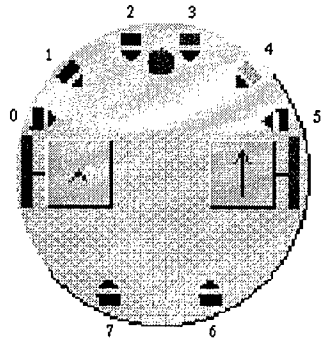


Figure 2: Simulated Khepera robot

To maximize the fit between real and simulated robot, the simulator takes into account noise at the level of a) sensor measurements (both light and distance) and b) motor commands (Olivier Michel, personal communication). Neural controllers trained with the Khepera simulator have been successfully transferred directly from the simulated to the real robot. In these experiments simulated and real robot showed very similar behaviour (using the same controller) with only minor quantitative differences (Michel, 1996).

3.3 Learning Algorithm

In all of the experiments discussed in this paper the control networks were trained with a modified version of the *complementary reinforcement backpropagation (CRBP) algorithm* (cf. Ackley & Littman, 1990, and Meeden, 1996). CRBP allows to turn the abstract signals of 'reward' and 'punishment' as given during reinforcement learning into precise error measures as they are required by backpropagation learning.

The real-valued output of the network is interpreted as a *search vector S*. The binary *output vector O* is produced stochastically by interpreting S(i) (i.e. the i-th element of S) as the probability that O(i) takes on the value 1. In the experiments described here the *action vector A* that sets the motor output in some case is derived from O, in others from S, depending on whether or not the output is required to be binary. In any case, if action A is rewarded the error measure (A-S) is propagated back in order to 'push' the network towards this vector. In case that A is punished the network is pushed towards the complement of O by backpropagating the error measure ((1 - O) - S).

Another feature of CRBP is the use of different learning rates for reward and punishment. In the simulations described here a learning rate of 0.2 was used for reward and 0.3 for punishment. Further details on how reinforcement was given in the individual experiments will be given in the following.

It should be noted that the focus of this paper lies on the proposed architecture and its capacities rather than on the learning algorithm. For more complex learning tasks than those discussed here CRBP (or backpropagation in general) might prove insufficient, and will in that case be 'replaced' in future work.

3.4 Experiments

The following three experiments were carried out and will be described in detail in the following sections:

(1) Obstacle avoidance through discrete action selection, showing that the SARN can learn to acquire FSA behaviour.

(2) Obstacle avoidance through continuous-valued direct motor control, showing that this architecture can as well systematically develop and utilize continuous state spaces.

(3) Obstacle avoidance while periodically switching between light seeking and light avoidance, showing that the proposed architecture, through complex, continuous but structured, state spaces, can combine the continuity and flexibility typical for the dynamics approach with complex structure and control as typical for the algorithmic approach.

All experiments were carried out in the simple environment shown in figure 3. The light source however was only contained in the environment in experiment 3.

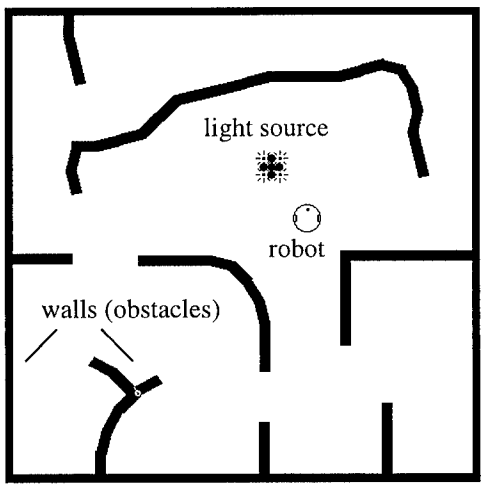


Figure 3: The robot's environment

3.4.1 Experiment 1: Discrete action selection

In this experiment the robot was trained on a rather simple task: to keep moving around in the environment (s. figure 3) while avoiding obstacles by selecting one out of three discrete actions (move forward, turn left, turn right) in each time step. At the motor speeds used here a 'move forward' moves the robot forward for about 5% of its own diameter while a 'turn' results in a turn (on the spot) of about 12 degrees in the respective direction.

Hence, the control network for this experiment had three output units, one for each action. The maximum output value determined the selected action and thereby the motor output, and the respective elements of output vector O and action vector A were set to 1 in any case whereas the remaining elements of O were determined according to the above scheme (cf. section 3.3) and the rest of A was set to 0. Furthermore the control network had three state units (as all following networks) and received as input the values of the six distance sensors at front and sides (i.e. sensors 0 to 5, normalized to values between 0 (no obstacle 'in sight') and 1 (touching obstacle))

Reinforcement was given as follows: At each time step the 'sensor-state' was determined by calculating the sum of those distance sensor readings, that exceeded a limit of 20% of the maximum value, multiplied with specific weights for the individual sensors. These weights were [1,2,5,5,2,1] for sensors 0 to 5, i.e. the sensors at the front were weighted higher than those at the side. The controller was 'punished' whenever its sensor-state value exceeded the last step's value, i.e. when the robot came too close to an obstacle or turned in the wrong direction (i.e. such that the measurements at the front increased). When there was no obstacle 'in sight' the controller was punished whenever it did not move forward.

Using this reinforcement scheme the robot learned to avoid punishment rather quickly. After about 10000 steps the ratio of punished steps sank below 5%. This simulation was repeated several times with similar results. Analysis of the resulting state spaces showed that in all cases the networks learned to approximate FSA with a rather small number of states (4 to 10). It should however be noted that, unlike in the algorithmic approach, this structure was acquired from reinforcement, i.e. as a result of the robot's own experiences, i.e. the robot has (at least partly) formed its own principles of behaviour.

Analysis of the context network and resulting input weights reveals that the controller network in this experiment makes only very little use of the context network and input weights only vary very slightly for different states. In fact, this experiment was repeated without context network, i.e. with an SRN-like first order recurrent network, and very similar results were obtained.

Furthermore these simulations were repeated with simple three-layer feed-forward networks of different hidden-layer sizes. In more than 80% of the simulations the robot got 'stuck' in oscillations (e.g. repeatedly carrying out left and right turns in a corner) within the first 5000 steps. The same behaviour was observed in experiments 2 and 3, due to which feed-forward networks have been excluded from further analysis in this paper.

3.4.2 Experiment 2: Direct motor control

In this experiment the robot's task basically was the same as before, to move around the environment (cf. figure 3) while avoiding obstacles. The control network's task however was significantly complicated: the real-valued network output was now used directly to set the speeds for the two motors (i.e. A takes on the values of S (cf. section 3.3)). That means the control network now had two output units, one for each motor, outputs of 0 were interpreted as maximum speed backward, 1 was interpreted as maximum speed forward, values in between resulted in corresponding motion at lower speed. The network still had three state units and used the same input values as before.

Reinforcement was given in the same way as before, except that as long as there was no obstacle in sight the robot was punished whenever at least one of the motors did not move forward at at least 80% of the maximum speed. In that case both elements of O were set to 0, i.e. the robot was reinforced to move forward at maximum speed.

Again, this experiment was carried out several times with SARNs, SRNs, and additionally Jordan networks (Jordan, 1986) (a Jordan network (JN) is similar to an SRN, except that in the JN it is the output units, in the SRN the hidden units whose values are used as extra input in the next time step). In most cases it took the robot longer to learn this task than in the previous experiment. Moreover the ratio of punished steps remained higher than before, which was mostly due to the fact that the robot tended to get closer to the obstacles since it did not turn on the spot immediately when detecting an obstacle. Instead its turns continuously got sharper the closer it got to the wall, such that it did avoid collisions, but often passed the 20%-limit. Figure 4 shows the average performance over the first 20000 time steps for 10 SARNs compared to 10 SRNs and 10 JNs. The SRNs and JNs used here had 3 hidden units, no significant improvement in performance was achieved with larger hidden layers. Here the SARNs typically learned faster in the beginning and converged to a lower degree of punishment (about 10%) than SRNs and JNs which showed rather similar performance (punishment ratio of about 16%/18%).

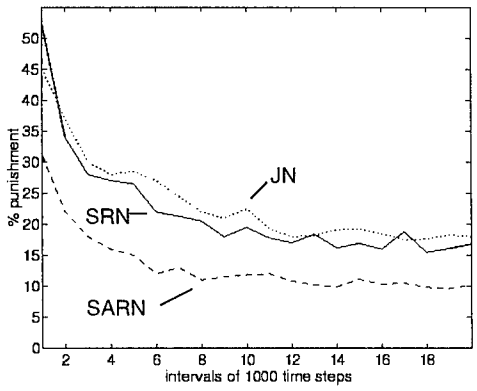


Figure 4: Convergence history for SARNs, SRNs & JNs

Figure 5 shows a typical SARN state space over 5000 time steps which reveals that the controller network no longer acted like a finite state automaton, but instead developed an almost continuous, but obviously constrained state space.

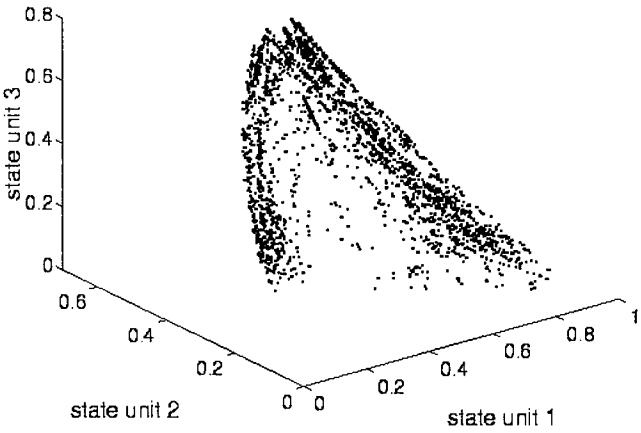


Figure 5: SARN state space - experiment 2

The relation between different contexts/situations and internal states is illustrated in figure 6, which shows point attractors (black circles) corresponding to the robot's most 'extreme actions', i.e. forward motion and the sharpest turns. These attractors are connected through basins of attraction (black arrows) through which the internal state evolves corresponding to the robot's motion within the environment. If for example the robot, while moving forward, encounters a wall on the left the input stimulus 'pushes' it into the (leftmost) basin of attraction towards a sharp right turn. I.e. the robot will first gradually increase its degree of turning until it either reaches the sharpest degree of turning or perceives the input stimulus ('wall on the left disappears') that pushes it over the 'border' (dashed line) into the neighbouring basin of attraction leading back to the state of moving forward. The gradual transition between moving forward and the points of sharpest turning allows the control network to adapt its turning behaviour to the required overall degree of turning, i.e. a 30 degree turn will result in a shorter trajectory (never reaching a 'turn' attractor) than a 90 degree turn.

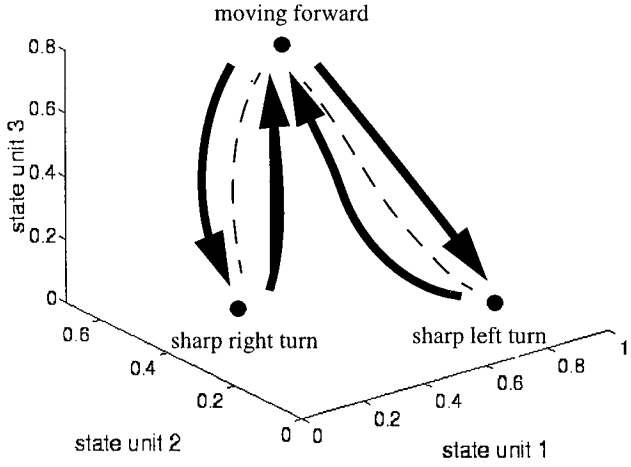


Figure 6: State space attractors and trajectories

Figure 6 and the matrix of biases and weights (white: positive, black: negative) from state to output layer (s. figure 7a) show that it is state unit 1 that determines the degree of turning (if turning at all). A high activation of state unit 1 results in a high speed for the right motor and a low speed for the left motor, i.e. in a turn to the left. A low activation for state unit 1, on the other hand leads to a low speed for the right motor and a high one for the left motor, i.e. to a turn to the right. From the state space we can further see that a medium value for state unit 1, obviously corresponding to a state of moving forward (cf. figure 6), always co-occurs with high activation for the other two state units (2 & 3) which leads to excitation of both output units, i.e. to high forward speeds for both motors.

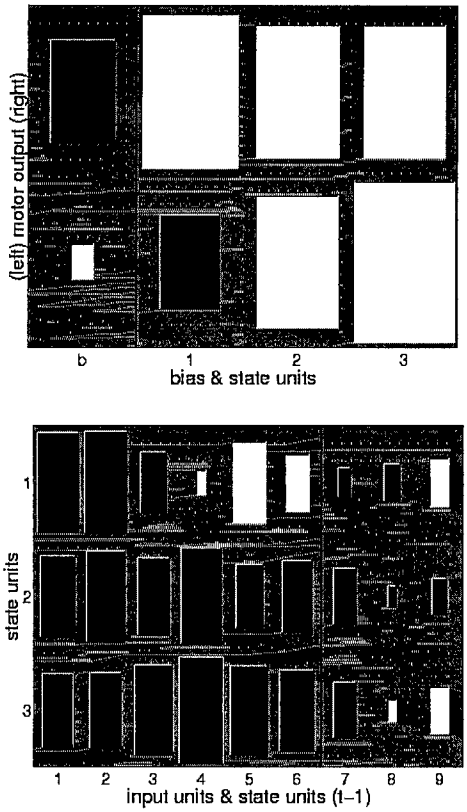


Figure 7: a) State-to-output weights, and b) Input-to-state weights (moving forward)

Figure 7b shows a typical input weight matrix (as set by the context network) for this state of moving forward. It can be seen that all sensor inputs have an inhibitory effect on both state unit 2 and 3. This means that the robot slows down as soon as it encounters an obstacle.

Figure 8 shows typical input weight matrices for the states of turning left and right which indicate that here the context network does make systematic, context-dependent use of its power of adapting the input weights (NB: the above output weight matrix (s. figure 8a) is of course context-independent (cf. figure 3)). In both cases all sensor inputs still have an inhibitory effect on state units 2 and 3. The weights for state unit 1 however, which 'balances' the direction of movement, are significantly different, since the network has to shift its 'attention' according to the direction that the input stimulus (causing the change of basin of attraction) has to come from. When turning left the positive weights for the right hand sensors are higher and the negative ones for the left hand sensors lower than before. This has the effect that, as long as there are obstacles on the right, state unit 1 will receive a high activation which, as explained above, leads to a sharp turn to the left. Analogous behaviour can be observed when turning to the right. Now the weights are set such that state unit 1 receives a low activation, resulting in a turn to the right, as long as there is an obstacle on the left.

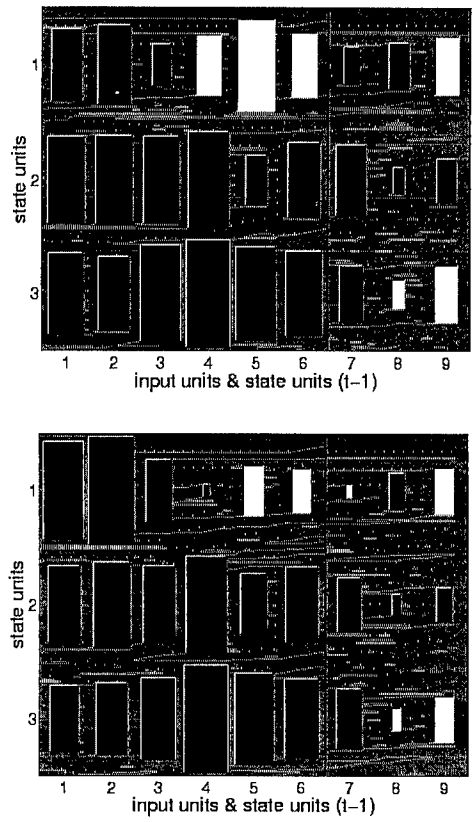


Figure 8: Input-to-state weights for a) turning left, b) turning right

It should be noted that the input weight matrices shown in figures 7 and 8 of course represent 'snapshots' from a continuum of states (cf. figures 6) and corresponding weights sets. The discussed tendencies and mechanisms have however been found to be generally valid, and the above state space (cf. figures 5 and 6) shows that the control network's internal state systematically reflects the robot's current context /situation.

As demonstrated above, the context network has obviously learned to exploit these regularities such that an adaptive weighting of inputs in the function network enables the control network to adapt the degree of turning to the current context/requirements. It has been observed in the majority of

simulations that with further training the differences between input weight sets, and thereby the ability to perform sharp turns (in both directions), increase with time. In the example discussed above this is realized by moving the two turning-attractors further towards lower values of state units 2 and 3 (i.e. towards sharper turns, cf. figure 6).

3.4.3 Experiment 3: Behaviour integration

The setup for experiment 3 was partially similar to that of the previous experiment. The motor speeds were again controlled directly by the control network. Part of the task was again obstacle avoidance and reinforcement for this part was given in the same way as before. The control network still had two output and three state units.

In addition to experiment 2, the environment (cf. figure 3) now also contained one 'big' light source (actually consisting of four light sources, combined with small obstacles to ensure that this light can be perceived as an obstacle too). Hence, the control network now also received input from light sensors 0 to 5 (normalized to values between 0 ('no light') and 1 ('maximum light value')), such that the network now had 12 sensory inputs (plus 3 units for the previous state). Furthermore the network received one extra input which periodically changed between 1 ('on') and 0 ('off'). In addition to the task of obstacle avoidance the robot should follow the light gradient towards the light whenever the extra input is 'on', but avoid the light otherwise. It should be noted that light can be sensed from a longer distance than obstacles, but cannot be perceived at all if there is an obstacle in the way between light sensor and light source.

In addition to the sensor-state value a light-state value was calculated in the same way in each time step (using the same weighting). As long as light was perceived at all the robot was punished for increasing its light-state value and rewarded for decreasing it as long as the extra input was 'off', and the other way round when the extra input was 'on'. As long as no light was perceived the robot should be moving around while avoiding obstacles, independent of the value of the extra input. In case of conflict between light-related behaviour and obstacle avoidance, light-related behaviour had the higher priority as long as light was perceived. That means that the robot, when the extra input is 'on', should actually hit a light source (or an obstacle close to it) as long as that helps to maximize its light-state value.

As before, this experimental setup was used to train several control networks. In most cases it took the robot equally long to learn the obstacle avoidance part of the task as in the previous experiment, and the ratio of punishments for getting close to obstacles sank to a similar level. The time it took the control network to learn the light-related behaviour varied between about 10000 and 40000 steps, which was probably mainly due to the fact that (a) the environment contained only one light source such that the robot could wander around for quite a long time without ever encountering light, and (b) the extra input was only 'on' 20% of the time (cycles of 200 time steps 'on' followed by 800 time steps 'off').

After that time however the SARN controllers did in most cases (8 out of 12) converge to an 'acceptable' level of punishment within 50000 steps, i.e. (a) the ratio of obstacle-related punishment sank below 10% (note that this kind of punishment is impossible while following the light) and (b) the ratio of punishment for incorrectly seeking/avoiding the light sank below 5% overall and below 20% while encountering the light one way or the other.

Again SRN and JN controllers, when trained on the same task, converged slower (cf. figure 4) to a significantly worse level of performance (for both network types the level(s) of punishment remained almost twice as high as the above numbers). This was mainly due to the fact that SRNs and JNs did not learn to switch between light avoidance and following, but instead kept periodically re-learning one and 'forgetting' the other, since they could not be integrated easily.

The behaviour acquired by the SARN controllers could typically be described as the following 'strategy':

(1) as long as no light is perceived: avoid obstacles, independent of the extra input, by turning as soon an obstacle is encountered,

(2) if extra input 'off' and light perceived: turn away from the light and move on,

(3) if extra input 'on' and light perceived: turn towards the maximum light perception and then move forward and hit the light source.

As in experiment 2 the control networks developed continuous state spaces, which in fact now exhibited even more constrained and structured subspaces containing a theoretically infinite number of possible states (s. section 3.4.4 for further analysis). The state space over 5000 time steps for a typical controller network is shown in figure 9, which also illustrates the relation between the robot's different contexts/situations and the correspondingly overlapping/intersecting subspaces of its internal state space.

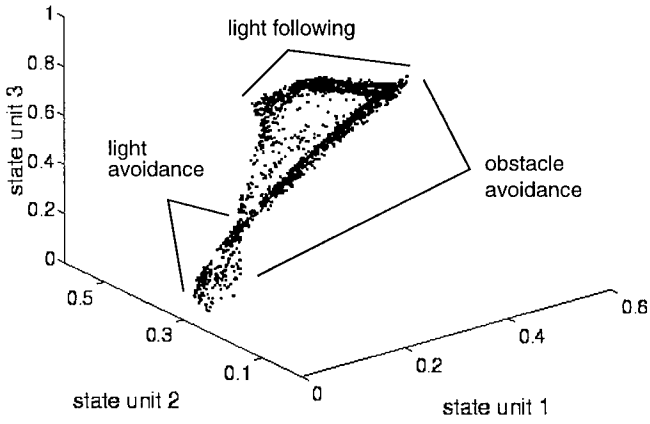


Figure 9: SARN state space - experiment 3

A look at typical state-to-output weights (s. figure 10a) reveals almost a 1:1 relation between state units and actions. A high activation for state unit 1 will inhibit the right motor output, i.e. probably lead to a right turn. Similarly a high activation of state unit 3 will result in a left turn. A high activation of state unit 2 alone will lead to a forward move.

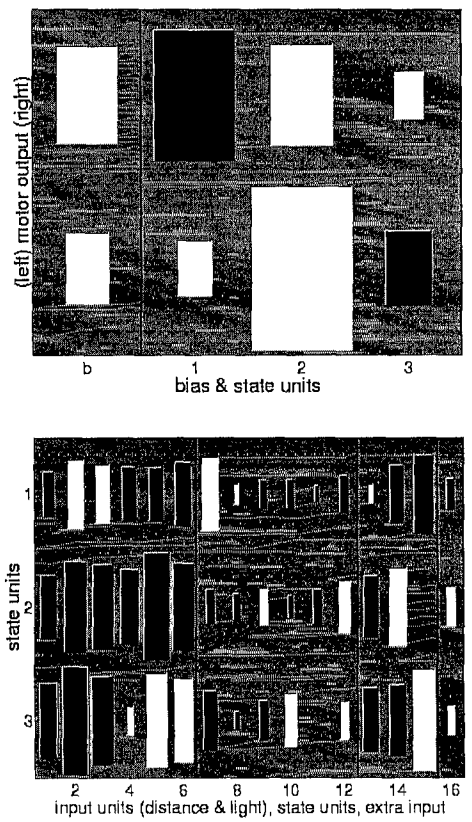


Figure 10: a) State-to-output weights, and b) Input-to-state weights for obstacle avoidance

Figure 10b shows input weights while performing obstacle avoidance (without light). It can be seen that the context network has learned to set the weights for distance sensors (first six units) significantly higher than those for light sensors (units 7 to 12). Encountering an obstacle leads to a low activation of state unit 2, which, as shown above, is responsible for moving forward. Depending on strength and location of the sensor readings state units 1 and/or 3 are activated which results in turning or possibly moving backwards. It has been found that turning by these networks is handled in a similar way as in experiment 2 (illustrated in the previous section), i.e. through a (slight) variation of a subset of the input weights.

The integration of obstacle avoidance and light following/avoidance however requires much more diverse weights. Nevertheless the following input weight sets (again, representative 'snapshots' for continuous groups of input weight sets) show that the control network has successfully learned to handle this problem exploiting the possibility of weight adaptation through the context network. A set of weights almost completely different from that for obstacle avoidance (cf. figure 10b) is used for light following (s. figure 11a). Now the weights for the light sensors have become relatively high whereas those for the distance sensors are now lower than those for 'pure' obstacle avoidance (cf. figure 10b). The sub-matrix of light sensor weights looks very similar to a negative copy of the distance sensor weight matrix, i.e. obstacle avoidance and light following make very similar use of the state units, but with opposite effects/directions. It can also be seen that the light sensor weights are larger than distance sensor weights, which corresponds to the fact that light-following should override obstacle avoidance.

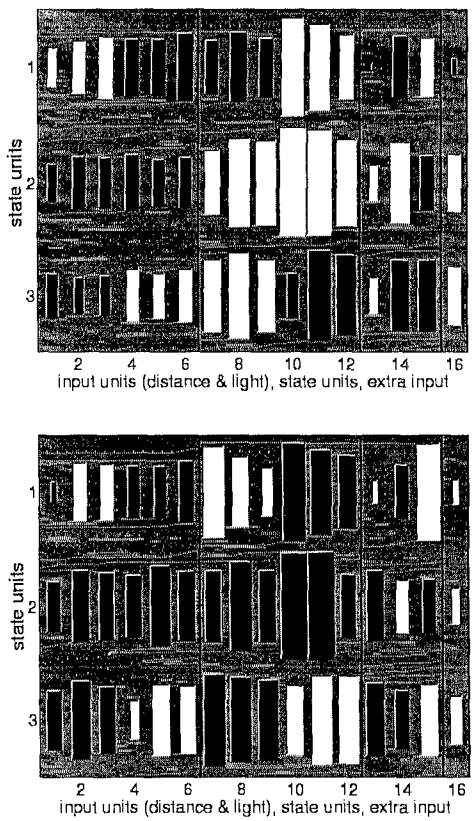


Figure 11: Input-to-state weights for a) light following, and b) light & obstacle avoidance

Finally, when combining light and obstacle avoidance (s. figure 11b), light and distance sensor weights are of similar magnitude, and they make almost exactly the same use of the state units' effect on the motor outputs.

The above analysis and results show that the architecture used here is capable of developing/exhibiting

(a) complex, continuous but structured, 'infinite state' spaces with systematically related weights,

(b) radically different input weight sets, corresponding to diverse contexts,

(c) a complex integration of different behaviour systems

that goes beyond a mere combination of effects at the level of actions as common in dynamics architectures.

3.4.4 Dimensional analysis of state spaces

The attractor regions in the SARN state spaces for experiments 2 and 3 (cf. figures 5 and 9) are similar to the structures found by Pollack (1990, 1991) in the state spaces of SCNs trained on formal language recognition. In both cases these regions contain a theoretically infinite number of states which however are not arbitrary or random but constrained by the mathematical principle of the underlying network dynamics (cf. Kolen, 1994a), i.e. they are so-called 'fractal' or 'strange' attractors to which there is no direct analogy in finite automata theory (cf. Pollack, 1990). Therefore they are here referred to as 'infinite state automata'.

The fractal dimension, a measure for the density with which a set of points covers a metric space, has been calculated for these 'infinite state automata' in order to validate and quantify the (subjective) impression that in the experiments reported here their complexity grew with the complexity of the learning task. Dimensional analysis of the internal states exhibited in experiment 2 (cf. figure 5) revealed a fractal dimension of about 1.4 (varying between 1.3 and 1.6 for different networks)[1], similar to the values found by Pollack (1991). For the internal SARN states developed in experiment 3 (cf. figure 9) the fractal dimension was found to be about 1.8 (between 1.6 and 1.9). These values indicate infinitely structured sets of internal states (Mandelbrot, 1983, cf. also Kolen, 1994a).

4 Discussion & Conclusion

This paper started with a brief review of the benefits and drawbacks of the major approaches to the design and implementation of behaviour systems, which could be briefly summarized as follows:

Complexity & structure: The algorithmic approach is capable of handling functionally & structurally complex behaviour, e.g. behavioural hierarchies, through a form of modularisation. The dynamics approach has no difficulties with the handling/integration of functional complexity, but with structural complexity beyond a simple cooperation of behaviours. Connectionist approaches have problems with functional complexity, do however offer distributed resources.

Continuity: Both dynamical and connectionist systems can utilize continuous state spaces, typically resulting in smooth and flexible behaviour. The algorithmic approach , due to its inherent commitment to modularity and FSA, typically can only handle finite, discrete states.

Adaptation & learning: Both dynamical and algorithmic systems are typically 'designed', and cannot adapt or extend their structure autonomously. Simple functional adaptation (of quantities, variables, etc.) in dynamical systems is possible. The connectionist framework incorporates learning mechanisms both learning 'from scratch' and continual adaptation are possible.

A self-adapting recurrent network architecture for the control of autonomous systems has been introduced in this paper. The capabilities of this architecture, which uses second order recurrent connections to dynamically compute its own sensory input weights, have been demonstrated and analysed in a number of experiments. It has been shown that

(1) the SARNs are capable of closely approximating FSA behaviour (experiment 1),

(2) the control networks have learned to establish a systematic (and continuous) relation between the robot's current context/situation and its internal state (as illustrated in detail for experiment 2),

(3) the control networks have acquired a systematic mapping of internal states to sensory input weights, which allows a dynamic adaptation to different contexts (through, if necessary, radical changes of input weights through the context network) (as illustrated in detail for experiment 3),

(4) the SARNs (in experiments 2 and 3) during learning have autonomously developed into 'infinite state automata' with internal state spaces whose complexity and structure (seem to) reflect complexity and structure of the learned task (cf. section 3.44),

(5) systematic 'infinite state' behaviour within continuous state spaces with a corresponding dynamic adaptation of the function network enables the overall system to exhibit smooth and continuous, but structurally complex behaviour.

It does of course remain an open question to which extent the architecture discussed in this paper can scale up to the integration of larger or more diverse sets of behaviour systems than those used here, as it would be necessary for the realization of a complete autonomous agent. Apart from this point and the validation on a real robot, future work will include further investigation of

(a) the relation between the complexity/structure of the learning task and that of the induced automaton,

(b) the intuitively appealing idea of incremental learning realized through dynamic extension/adaptation of existing internal state spaces to changing requirements or growing new subspaces for new tasks (cf. section 3.4.2).

1. using Barnsley's 'Box-Counting-Theorem' (1993), determined experimentally over the range of 0.05 to 0.5

It has however been shown in this paper that the approach presented in this paper, at least to some extent, allows an integration of

(a) the capacity to systematically utilize continuous state spaces as typically found in dynamical systems (a capacity which has been hypothesised to be the core of "the versatility and robustness of animal behavior" (Beer, 1995)),

(b) the complex structure and control of behaviour typical for algorithmic approaches (while avoiding the drawbacks of 'modularity by design'), and

(c) the self-organizing learning capacity of connectionist systems, combined with

(d) a simple, but powerful mechanism for context-dependent behaviour adaptation in individual agents,

i.e. an integration that combines the major strenghts of the invidual approaches while overcoming some of their major drawbacks.

Hence, the approach presented here, although far from fully evaluated yet, can be considered an interesting route for further work towards the synthesis of adaptive behaviour in truly autonomous agents.

References

Ackley, D. H. & Littman, M. L. (1990) Generalization and scaling in reinforcement learning, in Touretzky, D. S. (ed.) *Advances in Neural Information Processing Systems*, 550-557, Morgan Kaufmann, San Mateo, CA

Barnsley, M. F. (1993) *Fractals Everywhere*, second edition, Academic Press, San Diego, CA

Beer, Randall D. (1995) A dynamical systems perspective on agent-environment interaction, *Artificial Intelligence*, **72**, 173-215

Brooks, Rodney (1986) A robust layered control system for a mobile robot, *IEEE Journal of Robotics and Automation*, **2 (1)**, 1986, 14-23

Elman, J. (1990) Finding Structure in Time, *Cognitive Science*, **14**, 179-192

Kolen, John F. (1994a) Recurrent networks: State machines or iterated function systems?, in Mozer, M., Smolensky, P., Touretzky, D., Elman, J. & Weigend, A. (eds.) *Proceedings of the 1993 Connectionist Models Summer School*, 203-210, Erlbaum, Hillsdale, NJ

Kolen, John F. (1994b) *Exploring the computational capabilities of recurrent neural networks*, Ph.D. Dissertation, The Ohio State University

Jordan, M. (1986) Attractor dynamics and parallelism in a connectionist sequential machine, *Proceedings of the 12th Annual Conference of the Cognitive Science Society*, 531-546, Erlbaum, Hillsdale, NJ

Mandelbrot, B. (1983) *The Fractal Geometry of Nature*, Freeman, San Francisco, CA

Meeden, Lisa (1996) An Incremental Approach to Developing Intelligent Neural Network Controllers for Robots, to appear in *IEEE Transactions on Systems, Man, and Cybernetics*, **26**, special issue on Learning Autonomous Robots

Michel, Olivier (1995) Khepera Simulator version 1.0 - User Manual, University of Nice-Sophia Antipolis, Valbonne, France, (http://wwwi3s.unice.fr/~om/khep-sim.html)

Michel, Olivier (1996) An Artificial Life Approach for the Synthesis of Autonomous Agents, in *Artificial Evolution - AE95*, Springer Verlag

Mondada, F., Franzi, E., & Ienne, P. (1993) Mobile robot miniaturisation: A tool for investigation in control algorithms, in *Third International Symposium on Experimental Robotics*, Kyoto, Japan

Pollack, Jordan B. (1990) Language Acquisition via Strange Automata, *Proceedings of the 12th Annual Conference of the Cognitive Science Society*, 678-685

Pollack, Jordan B. (1991) The induction of dynamical recognizers, *Machine Learning*, **7**, 227-252

Steels, Luc (1993) Building agents with autonomous behaviour systems, in Steels, L. & Brooks, R. (eds.) The 'artificial life' route to 'artificial intelligence'. Building situated embodied agents, Lawrence Erlbaum, Hillsdale, NJ

Steels, Luc (1994) The artificial life roots of artificial intelligence, *Artificial Life*, **1**, 75-100, MIT Press, Cambridge

Steels, Luc (1995) When are robots intelligent autonomous agents?, *Robotics and Autonomous Systems*

Tani, Jun (1996) Model-Based Learning for Mobile Robot Navigation from the Dynamical Systems Perspective, to appear in *IEEE Transactions on Systems, Man, and Cybernetics*, **26 (3)**, special issue on Learning Autonomous Robots

Centrally-generated and reflexive control strategies in the adaptive behavior of real and simulated animals

Jim H. Belanger and Mark A. Willis

Arizona Research Laboratories, Division of Neurobiology
University of Arizona
PO Box 210077
Tucson, AZ 85721-0077 USA
jimb@neurobio.arizona.edu, maw@neurobio.arizona.edu

Abstract: In our studies of the odor-guided navigation of real and simulated insects we have explicitly addressed the strengths and weaknesses of reflexive and centrally patterned control strategies in successfully adapting to a complex environment. While often presented as contesting organizational principles, reflexive and centrally-patterned behaviors actually exist on a continuum ranging from purely reflexive control with no central contribution to purely centrally-patterned control with no sensory input. Researchers who use biological systems as a take-off point for animat design need to address this continuum and consider what combination of reflexive and centrally-patterned control might prove most adaptive, given a specific task and environmental complexity. Results from our neuroethological studies suggest that the flight behavior of real moths represents a centrally organized search strategy that continuously exposes the animal to sensory information that is critical to both locating the odor source, and to maintaining an appropriate relationship between the environment and the sensori-motor system of the moth (*i.e.*, dynamic sensation).

Introduction

There has been a great deal of success in using principles inspired by biology to design animats that display adaptive behavior (*e.g.*, Brooks 1986; Beer 1990; Maes 1991). As this field moves out of the phase of 'existence proofs' and 'feasibility prototypes', however, it may be time to look more carefully at the specific biological inspirations for these designs. In particular, it is worth considering to what extent animat designs are, or should be, based on centrally-generated *versus* reflexive behaviors. Our own work, using real and simulated animals to study the adaptive control of odor-guided locomotion (Belanger and Willis 1996), has raised interesting questions about the relative merits of both designs. Obviously, there are few, if indeed any, situations in which a design will be completely one or the other. But there may be circumstances in which it is useful to bias control systems in either direction. By looking at biological solutions to adaptive control problems, we may be able to infer general principles about what those conditions, if any, may be.

An important consideration is deciding what should be the proper level of analysis for 'adaptiveness'. In natural selection, adaptiveness is measured at the level of the organism/environment interface. No structure can have adaptive value without reference to the environment in which it must exist. Thus, the biological adaptive systems that we study are optimized (in some sense) around survival/reproduction in a given environment, and the distributed nature of the system means that the implementation of that 'optimization' must also be distributed. This issue is important, because we may unintentionally build constraints into animats by designing them around specific architectures chosen for hypothesized (rather than real) optimization targets.

In the following discussion, we examine the nature of central and reflexive control strategies from the perspectives of both motor output and sensory input . (The former has a rich biological literature, while the latter is actually being driven largely by robotics research.) Our aim is twofold: first, to share our views, as neuroethologists, on this issue in general, and second, to offer the insights that we are deriving from our studies of a specific biological problem in adaptive behavior.

Generation of rhythmic behaviors in animals - A historical perspective

Neurobiologists have been studying the mechanisms underlying rhythmic behaviors for more than a century. A major reason for this is practicality. Due to their repetitive nature, rhythmic behaviors are easier to elicit, perturb, and examine than are episodic behaviors. From the animat perspective, this work is important because a basic task required of autonomous agents is locomotion, generally a

rhythmic act. (Note that there is no *a priori* reason why the locomotor behavior of an animat must be rhythmic - numerous arhythmic locomotory systems could probably be constructed. Designing the locomotor structures of animats using rhythmic designs is an affirmation of our belief that natural selection has produced efficient solutions to basic problems.) However, a historical perspective on this literature is valuable in understanding some of the current ideas and controversies regarding the generation of rhythmic behaviors. For that reason, we will sketch (in **very** broad strokes) some of the important points in the literature, and how they have contributed to our present understanding. Our coverage is necessarily brief, and the reader is directed to the numerous reviews cited for more thorough coverage of particular subjects.

In 1906, Sir Charles Sherrington published *The Integrative Action of the Nervous System,* in which he outlined his ideas on the reflex as the fundamental unit of behavior. He reported work on cats whose functional central nervous system (CNS) had been reduced to just the spinal cord, showing that they assumed characteristic postures, with the limbs held relatively stiff and outstretched. More interestingly, diagonally-opposite limbs appeared to be functionally coupled, such that an imposed movement swinging a given limb forward caused its counterpart to move similarly, while the remaining pair of limbs would simultaneously swing backwards. These so-called 'reflex figures', in the complete absence of any input from higher centers, suggested a mechanism by which the nervous system could generate rhythmic behaviors such as walking, flying or swimming: If each phase of a particular movement produced sensory feedback which reflexively initiated the next phase, then this set of 'chained reflexes' would be sufficient to produce locomotion. (Starting and stopping the process were obvious sticking points in this hypothesis, but these were supposed functions of the aforementioned 'higher centers'.) At the time, this hypothesis would have been very difficult to test experimentally, were it not for an organizational peculiarity of the vertebrate spinal cord: Almost all sensory input enters the spinal cord by a path which is anatomically distinct from that by which motor output exits. This allowed Graham Brown (1911) to selectively eliminate virtually all sensory input, and to demonstrate that animals so treated could still produce relatively normal locomotion. These experiments led to the first explicit proposal for a central pattern generator (CPG)(Brown 1914). In contrast to the chained reflex model, Brown proposed that there was a circuit or circuits within the spinal cord which could produce the motor pattern(s) for walking in the complete absence of sensory feedback. His 'half-center' model was based on reciprocal inhibition between two populations of neurons. Provided that these two populations received tonic excitatory input, the two populations would alternate in producing output, resulting in a two-phase rhythmic pattern. (Note the fact that this model deals with the starting and stopping problem implicitly.) Subsequently, Brown's initial concept has been generalized to give the definition of a CPG: a circuit of neurons which can produce rhythmic output in the absence of any rhythmic input. Importantly, this definition does not deny that sensory input has a role in shaping the output of the circuit, merely that it is not **required** for rhythmic output. Indeed, in the face of Brown's experiments, Sherrington accepted that the basic rhythmicity of locomotion was produced centrally, and that reflexes functioned to adjust this fundamental rhythm according to "the exigencies of the environment" (Liddell and Sherrington 1924).

This should have signaled the beginning of a fruitful period of research aimed at understanding the relative contributions of central and reflexive control systems. The actual outcome, however, was that for several decades research was aimed at supporting either the CPG or the peripheral control model (see the thoughtful review by Delcomyn 1980). As Grillner (1985) and others have pointed out, this unfortunate - and unnecessary - polarity deflected attention away from the actual mechanisms underlying pattern generation. The fundamental question of whether central oscillators really even exist was not put to rest until Donald Wilson (1961) showed that the completely isolated CNS of locusts was capable of producing motor output that was qualitatively similar to that seen in the intact animal: activity in motoneurons supplying the wing elevator and depressor muscles showed the proper pattern of out-of-phase bursting required to drive the wings during flight. This experiment quelled the persistent criticism of experiments such as Brown's (1911): How could the experimenter be sure that **all** sensory inputs had been removed? In Wilson's preparations, there was no sensory input because there were no sense organs - just an isolated CNS in a dish. This isolated CNS paradigm became the touchstone for demonstration of a CPG underlying a given behavior. If an isolated nervous system could produce a pattern of motor outputs that was qualitatively similar to that seen in the intact animal, then a CPG was presumed to underlie the behavior in question. Subsequently, CPGs associated with rhythmic behaviors such as walking, swimming, flying, breathing and feeding were demonstrated in representatives from most of the major phyla (see reviews by Delcomyn 1980; Kristan 1980; Roberts and Roberts 1985; Grillner and Wallén 1985; Jacklet 1989).

With the existence of CPGs apparently well-established, workers began to analyze the mechanisms underlying their ability to generate rhythmic outputs. CPGs were broadly split into two categories: those in which the capacity to produce oscillations was an endogenous property of individual cells in the CPG, and those in which oscillation was an emergent property of the network of cells (Kristan 1980; Selverston and Moulins 1985). In fact, many (most?) CPGs are hybrids of these two basic types, in which the rhythmic output of the network is an emergent property of that network, but members of the network are also capable of endogenous oscillations (Getting 1989).

As mechanistic details of CPGs accumulated, it became clear that the CPG/chain reflex division is probably untenable. Perhaps the best indication of this came when it was demonstrated that the canonical CPG - the locust flight oscillator - actually is critically dependent on sensory input for its normal functioning (Wolf and Pearson 1987). There is definitely a central oscillator, which can function in the complete absence of any sensory input. The fundamental flight rhythm in the intact animal, however, is not produced by this circuit. In one sense, it is merely a semantic difficulty whether we choose to include sensory input in the definition of a CPG. Bässler (1986) has argued that "it is trivial that the CPG is only a fraction of the whole network (including sense organs) responsible for the timing cues of the rhythmic motor output in the *intact* animal". He suggests the use of the term 'pattern generator' for the combination of central networks and sensory inputs responsible for rhythmic behaviors. Under this view, to which we subscribe, the most useful approach is to regard CPGs and chained reflexes as two ends of a continuum, rather than as a dichotomy. Pattern generators underlying different behaviors can be placed at different points along the continuum, depending on the extent to which sensory input is used to generate the basic rhythm. The goal for animat designers then becomes one of determining whether there are adaptive reasons why mechanisms underlying a particular behavior occupy a particular point along the continuum. (*Caveat scientist* - the reasons why a particular type of pattern generator underlies a particular behavior may have as much to do with the organism's evolutionary history as with current adaptive value (Dumont and Robertson 1986)).

As an example of an adaptive argument, consider the types of pattern generators proposed to underlie insect walking. Most of the literature in this area has focussed on animals adapted to two very different lifestyles: the fast-moving american cockroach (*Periplaneta americana*) and the relatively glacial stick insect (*Carausius morosus*). The pattern generator underlying walking in the stick insect appears to be almost a pure set of chained reflexes, with virtually no evidence of a CPG (Bässler and Wegner 1983). (Although recent work suggests the existence of central oscillators in the stick insect (Bässler 1993; Büschges *et al.* 1995), it is not yet clear what role these play in walking in intact animals.) The frequency range over which this pattern generator operates ranges from 0.5 to 4 Hz, and it generates statically-stable gaits over this entire range (Graham 1985). Cockroaches, in contrast, have stepping frequencies up to 27 Hz, and utilize statically-stable gaits only when running at low speeds (Full and Tu 1991). At the highest speeds, cockroaches use a bipedal gait, and are dynamically-, but not statically-, stable (Full and Tu 1991). Based on the physiological data, it is unclear where on the continuum the pattern generator for cockroach walking should be placed. Early work suggested the presence of a CPG (Pearson 1972), but more recent data suggest that the behaviors examined in the physiological experiments were struggling and righting reflexes, rather than normal walking (Zill 1986). Work on sensory reflexes in walking cockroaches indicates that sensory input is important in walking at slow speeds only (Zill and Moran 1980). In fact, it has been argued that insects running at high speeds might have to actively inhibit sensory input in order to prevent reflexes from disrupting their gait (Hoyle 1976). It thus appears that the cockroach pattern generator may shift its position on the sensory input continuum according to its output frequency, sliding towards the CPG end of the spectrum as the animal increases its speed. The stick insect, in the apparent absence of such dynamic control, may be restricted to slow gaits purely as a result of its pattern generator design. This has obvious potential implications for animats whose locomotor controllers are based on this design (*e.g.* Espenschied *et al.*, 1993).

Central generation of sensory behaviors - 'Dynamic sensation'

It seems obvious that most movements or behaviors, regardless of how they are activated, could result in the acquisition of sensory information critical for successful adaptation. It is also clear that many, if not all, animals actively perform certain behaviors to acquire such information. Indeed, it appears that some behaviors function specifically to "scan" for information of many modalities. While the idea of generating movement to acquire sensory input (*i.e.*, active perception) is not new to those working on animat design (Bajcsy 1993; Sandini et al., 1993), until now it has been applied almost exclusively to vision (Aloimonos 1993). This unnecessarily restrictive viewpoint seems to stem primarily from the applications and specific tasks to be performed by the animats in question. As autonomous agents are developed to perform specific tasks (*e.g.*, locate sources of toxic spills), or to inhabit environments where vision may not provide sufficient information (*e.g.*, deep undersea) the universal application of movement for sensation will become more obvious. A logical framework for biological information acquisition was presented recently at an National Science Foundation Workshop, entitled "Evolution of Biological Information Processing Systems" (Arbas, unpublished).

This framework comprises three main categories: 1) passive sensation, 2) active sensation, and 3) dynamic sensation. As one might expect, passive sensation describes the case in which environmental energy impinges on a static receptor(s) or animal (or animat). Active sensation has been used (above and Bajcsy 1993) to describe the case where animals or agents emit some form of energy into the environment, and specific sensory arrays with their underlying circuitry extract and process information based upon the modification of the signal by its interaction with the environment. The term dynamic sensation has been introduced to describe the situations where animals perform active behavior through which sensory information crucial for adaptive behavior may be extracted from the environment.

Those working on animat design have argued, with respect to vision, that movement for information acquisition should be lumped with active sensation as defined above (Balcsy 1993), and in fact entire volumes have been published organized around this idea (Aloimonos 1993). It is our feeling, especially with respect to the growing number of examples of "moving to sense" in animals (*i.e.*, the universality of this principle), that categorizing active sensing separately from dynamic sensing, as defined above, represents a real and fundamental difference in how the behaviors are organized and executed.

In animals, the active expression of movements to enhance or enable sensation has been reported for most sensory modalities. As in much of the animat work in this area, many of the best known examples of dynamic sensation center on the visual-motor system. Probably the best known example of this is the rocking and peering movement used by standing locusts to obtain information, through motion parallax, on jumping distance to targets (Collett 1978; Sobel 1990). Recent observations of nest building wasps and bees strongly suggest that flying insects may perform specific maneuvers to acquire information used in flight control and landmark learning (bees: Lehrer 1991, 1993; solitary-nesting wasps: Zeil 1993a,b; socially-nesting wasps: Collett and Lehrer 1993). In all of these cases an individual insect flies a stereotyped pattern upon leaving the hive or nest. It has been proposed that, embedded in this pattern may be specific visual sampling points, during which information on the orientation and location of the nest and nest entrance are acquired (Collett 1992; Collett and Lehrer 1993). In some cases, the points of visual information acquisition seem to be counterturns (at the ends of straight legs flown at an angle to the target), during which the head and longitudinal axis of the insect are aimed at the entrance to the nest. In contrast, the solitary nesting wasps studied by Zeil (1993 a,b) appeared to be memorizing elements of parallax fields surrounding major landmarks associated with the nest and nest entrance.

Visual flow field feedback generated by active behavior may be useful in tasks other than navigation. An elegant series of experiments on the optomotor control of flight in *Drosophila melanogaster* concluded that actively-generated fluctuations in torque result in the continuous re-calibration of the optomotor control system (Heisenberg and Wolf 1984; Wolf and Heisenberg 1990). Continual minor fluctuations in torque (even during extended periods of stable flight), typically alternating in direction, have been interpreted as the optomotor control system actively "feeling for" input. This optomotor control system enables the tethered fly to discriminate between self-generated movement and externally-generated movement in <50 ms. This type of periodic sampling via self-generated movement has been interpreted as being crucial to the ongoing stability of this visually controlled behavior (Wolf and Heisenberg 1990). A similar form of 'self-calibration' has also been proposed to underlie the visual flight stabilization system in locusts (Möhl 1989).

Inspired by flying insects, parallax and flow field cues have been used in the design of both a successful autonomous simulated flying agent and an autonomous flying robot. The simulated agent successfully used motion parallax cues to control its airspeed and altitude as it flew over a varying topography (Mura and Franceschini 1994). In a similar experiment, an autonomous robot was constructed with a visual control system inspired by the visual systems of flies. Using only simple movements designed to generate flow fields for the visual system, this robot was able to locomote through obstacles and locate a target without any internal representation of the environment (Franceschini *et al.* 1992). In fact, the authors of both of the above studies noted that "The tight reciprocal interactions which are exerted between the sensory block and the motor block suggest that there is a mutual reliance between the two tasks where 'acting in order to perceive' becomes just as important as 'perceiving in order to act' " (Mura and Franceschini 1994).

Similar scanning movements have been observed in animals navigating toward distant sources of auditory signals. In some species of tree frogs and crickets, females locate males for mating by orienting and locomoting toward them in response to their species specific-auditory call (Rheinlaender et al. 1979; Rheinlaender and Blatgen 1982). In each case the females move back-and-forth across a track that would take them directly to the sound source. The performance of this active behavior appears to carry the females out of a "zone of ambiguity" directly in front of the sound source, in which their auditory equipment is unable to determine if the sound source is to the left or right. In addition to this zigzag walking, female frogs often execute a stereotyped side-to-side head movement just prior to initiating locomotion toward the source. Jumps made after head scanning are oriented more directly toward the sound source than those without head scanning (Rheinlaender *et al.* 1979).

Olfaction too has associated active behaviors. Antennation of the ground in insects, or tongue flicking in reptiles, either to sample a chemical gradient (tropotaxis), or in a searching pattern to discover the location of a chemical signal through sequential samples (klinotaxis) during trail-following, are examples of dynamic sensation directed at the acquisition of olfactory information (Schöne 1984; Schwenk 1994). Lobsters actively flick their olfactory organs, the antennules, as a way of decreasing the depth of the boundary layer around the odor sensors and driving odor molecules into close contact with the sensors. In effect, this is "sniffing" in an aquatic environment (Schmidt and Ache 1979).

The animals that we study, the male sphinx moth, *Manduca sexta*, locate mates by flying up wind-borne plumes of sexual pheromones released by receptive, conspecific females (Arbas *et al.* 1993). Males generate a zigzagging upwind flight track

while they are flying in the pheromone plume. The moths utilize mechanosensory information and visual flow field cues to control their airspeed and altitude (with regard to the visual aspects, similar to the animats described above). Recent observations in our laboratory (Willis and Arbas 1996) suggest that male moths flying in pheromone plumes also execute specific maneuvers which may result in the acquisition of information used for flight stabilization and pheromone source localization. Information acquisition during such maneuvers requires simultaneous processing of several modalities of sensory input and minimally, the operation of a short term memory. Male moths are also able to regain contact with the pheromone plume if contact is lost due to wind shifts or a misguided maneuver. This is typically accomplished by the execution of a specific search strategy during which upwind progress ceases, ground speed increases, and flight tracks across the wind direction, with expanding lateral and vertical excursions, are executed. This behavior, termed "casting", appears to be an active strategy aimed at maximizing the likelihood of recontacting the odor plume.

Clearly, the ability to express sophisticated multi-component movements or sequences of movements that are stored "on board" has been critical to the adaptive behavior of actual organisms, particularly in dealing with complex sensory environments. In fact, as we will show below, our simulation experiments suggest that such a capability is essential to the successful behavior that we observe in moths tracking pheromone plumes.

Centrally-generated vs reflexive behaviors in a simulation of odor-oriented flight

Our work in neuroethology is aimed at understanding the neural mechanisms by which insects are able to locate specific resources (food or mates) by following odor plumes (see Arbas *et al.* 1993; Belanger and Willis 1996). A key element in the complexity of this task is that odor plumes in most natural environments are turbulent, and so concentration gradients contain little reliable information about the direction to the source of the odor. The behavior is multimodal, requiring the integration of olfactory, visual, and mechanoreceptive information with (perhaps) behavioral strategies designed to maintain contact with the odor plume. There is a large body of work dealing with insect orientation to odor sources, recent reviews of which may be found in Belanger and Willis (1996) and Cardé and Minks (1996).

Several different behavioral strategies have been proposed to explain the characteristic 'zig-zag' tracks produced by moths flying upwind in an odor plume. However, the complexity of the interactions between the moving animal and the turbulent structure of the plume makes it necessary to simulate explicitly the behavior of any proposed model in order to accurately test its predictions. To this end, we have developed a simulated wind tunnel to use as a 'virtual environment' in which to test behavioral models (Belanger and Arbas 1996). The simulator can replicate any experiment which can be performed with real moths. In keeping with our belief that the correct level for analysis of adaptive behavior is the interaction between the organism and the environment, the simulated wind tunnel reproduces the kinds of turbulence real moths encounter. It uses several types of 'odor plume', with which model moths can interact. The simulated plumes have characteristics that are indistinguishable from those of real plumes measured in a wind tunnel or in the field (Belanger and Willis 1996).

Based on hypotheses that have been presented in the literature, we constructed behavioral-level models of pheromone-modulated flight, and used the simulator to explore their performance (Belanger and Arbas 1996). All of the models are behavioral-level schemas (Cobas and Arbib 1992), implemented as augmented finite-state machines (Brooks 1986). The models vary considerably in their relative emphases on reflexive *versus* centrally-generated behavior. All of the models that we have implemented use mechanosensory and visual input to maintain a constant course relative to the wind (as real moths are proposed to do (Ludlow 1984)). As far as is possible, we have constrained the models using biological data (see Belanger and Arbas 1996 for details of the models). Thus there are realistic delays imposed between the animal's physical encounter with pheromone in the air and the animal's perception of the odor, and likewise between any perception of odor and the subsequent behavioral response.

The simplest models are purely reflexive, and use olfactory information to modify their behavior. *Pheromone-On anemotaxis* models respond to the detection of pheromone by resetting their flight control system to stabilize on a new course, the only constraint of which is that it not be downwind. *Pheromone-Off anemotaxis* models respond similarly, but only when an ongoing pheromone signal is lost. Surprisingly, these two very simple models are moderately successful at solving the problem of tracking a pheromone plume for the 2 m length of our standard simulated wind tunnel. For biologically-reasonable settings of the various parameters in the models, they succeed in locating the odor source around 30% of the time. However, this value is only attained if the odor plume is fairly homogeneous (corresponding to odor plumes found at low wind speeds). As turbulence breaks up the structure of the odor plume, these purely reflexive models fail, because they encounter gaps in the odor plume. Thus the success rate is around 10% when these models attempt to track plumes with significant gaps. Real moths, under equivalent conditions in our behavioral wind tunnel, have a success rate of about 90%.

Two somewhat more complicated models, *Counterturner* and *Surger*, are implementations of hypotheses which have been proposed recently to explain the orientation of moths to

odor plumes (see Cardé and Minks 1996). Each of them incorporates the centrally-generated behavior - 'casting' - which real moths appear to produce upon loss of an odor signal. The two models differ in their response to the detection of pheromone. *Surger* reacts to each pheromone event with a reflexive, upwind-oriented flight segment of fixed duration - essentially a ballistic reflex triggered by contact with pheromone. *Counterturner* responds to pheromone by turning on a centrally-generated series of upwind-oriented turns, which continues for as long as pheromone is being detected. The essential difference between the two is that pheromone input is permissive for *Counterturner*, whereas it is instructive for *Surger*. Considered another way, *Counterturner* represents a switching between two sets of centrally-generated behaviors, while *Surger* is either a series of chained reflexes, or a centrally-generated behavior, depending on whether the animal is within the pheromone plume or not. Both models achieve success rates of up to 40% (still well below real moths), and neither is affected by the degree of turbulence present in the environment. *Surger*, however, is very sensitive to the settings of a number of behaviorally-important parameters in the model (such as the latency to respond to pheromone, or the duration of the upwind surge). *Counterturner,* in contrast, is quite robust to changes in model parameters.

The one solid conclusion from our simulations so far is that none of the currently-proposed models is sufficient to account for the success that real moths show in tracking odor plumes. Therefore, our next generation of models will incorporate our evolving ideas on dynamic sensation, to investigate how such strategies affect the success of the models. It is still early in our simulation studies to draw solid conclusions which we can generalize to other systems. Tentatively, we can suggest that centrally-generated strategies appear to be necessary to cope with complex or heterogeneous environments. Centrally-generated strategies also seem to be more robust to changes in their parameters than do reflexive strategies.

Conclusions - Considerations for animat designers

1. There appear to be some general conditions under which either centrally-generated or reflexive control designs are likely to be superior.

2. Based on the rhythmic behavior literature, sensory input appears to be a fundamental part of pattern generator circuits under most circumstances. Exceptions occur when the cycle periods of the rhythm generators are too short to allow for transmission and processing of sensory information (*e.g.* cockroach running).

3. Centrally-generated behaviors seem to be essential for extracting sufficient sensory information from complex or heterogeneous environments.

Acknowledgments: This work was supported by NSF grant IBN9216532 (EAA and MAW) and an NSERC postdoctoral fellowship (JHB).

References

Aloimonos Y (1993) *Active Perception.* Lawrence Erlbaum Associates: Hillsdale.

Arbas EA, Willis MA, and Kanzaki R (1993) Organization of goal-oriented locomotion: Pheromone-modulated flight behavior of moths. In, *Biological Neural Networks in Invertebrate Neuroethology and Robotics.* Beer RD, Ritzmann RE, and McKenna T, eds. Academic Press: San Diego.

Bajcsy R (1993) Active perception and exploratory robots. In, *Robots and biological systems: Towards a new bionics.* Dario P, Sandini G, and Aebsicher P, eds. Springer-Verlag: Berlin.

Bässler U (1986) On the definition of central pattern generator and it sensory control. *Biol Cybern* **54**: 65-69.

Bässler U (1993) The walking- (and searching-) pattern generator of stick insects, a modular system composed of reflex chains and endogenous oscillators. *Biol Cybern* **69**: 305-317.

Bässler U, and Wegner U (1983) Motor output of the denervated ventral nerve cord in the stick insect *Carausius morosus*. *J Exp Biol* **105**: 127-145.

Beer RD (1990) *Intelligence as Adaptive Behavior: An Experiment in Computational Neuroethology.* Academic Press: San Diego.

Belanger JH, and Arbas EA (1996) Behavioral strategies underlying pheromone-modulated flight in the moth *Manduca sexta*: Lessons from simulation studies. Submitted manuscript.

Belanger JH, and Willis MA (1996) Adaptive control of odor-guided locomotion: Behavioral flexibility as an antidote to environmental unpredictability. *Adapt Behav*: in press.

Borst A, Egelhaaf M (1989) Principles of visual motion detection. *Trends in Neurosci* **12**: 297-306.

Brooks RA (1986) A robust layered control system for a mobile robot. *IEEE J Robot Automat* **RA-2**:14-23.

Brown TG (1911) The intrinsic factors in the act of progression in the mammal. *Proc Roy Soc Lond Ser B* **84**: 308-319.

Brown TG (1914) On the nature of the fundamental activity of the nervous centres; together with an analysis of the conditioning of rhythmic activity in the act of progression, and a theory of the evolution of function in the nervous system. *J Physiol* **48**: 18-46.

Büschges A, Schmitz J, and Bässler U (1995) Rhythmic patterns in the thoracic nerve cord of the stick insect induced by pilocarpine. *J Exp Biol* **198**: 435-456.

Cardé RT, and Minks AK (1996) *Pheromone Research: New Directions.* Chapman and Hall: London.

Cobas A, and Arbib M (1992) Prey-catching and predator avoidance in frog and toad: Defining the schemas. *J Theor*

Biol **157**: 271-304.

Collett TS (1978) Peering-a locust behaviour pattern for obtaining motion parallax information. *J Exp Biol* **76**: 237-241.

Collett TS (1992) Landmark learning and guidance in insects. *Phil Trans R Soc B* **337**:295-303.

Collett TS, and Lehrer M (1993) Looking and learning: a spatial pattern in the orientation flight of the wasp Vespula vulgaris. *Proc R Soc Lond B* **252**:129-134.

Delcomyn F (1980) Neural basis of rhythmic behavior in animals. *Science* **210**: 492-498.

Dumont JPC, and Robertson RM (1986) Neuronal circuits: An evolutionary perspective. *Science* **233**: 849-853.

Espenschied KS, Quinn RD, Chiel HJ, and Beer RD (1993) Leg coordination mechanisms in stick insect applied to hexapod locomotion. *Adapt Behav* **1**: 455-467.

Franceschini N, Pichon JM, and Blanes C (1992) From insect vision to robot vision. *Phil Trans R Soc Lond B* **337**:283-294.

Full RJ, and Tu MS (1991) Mechanics of a rapid running insect: Two-, four- and six-legged locomotion. *J Exp Biol* **156**: 215-231.

Getting PA (1989) Emerging principles governing the operation of neural networks. *Ann Rev Neurosci* **12**: 185-204.

Graham D (1985) Pattern and control of walking in insects. In, *Advances in Insect Physiol* **18**: 31-140.

Green PR, Davies MNO, Thorpe PH (1994) Head-bobbing and head orientation during landing flights of pigeons. *J Comp Physiol A* **174**:249-256.

Grillner S (1985) Neurobiological bases of rhythmic motor acts in vertebrates. *Science* **228**: 143-149.

Grillner S, and Wallén P (1985) Central pattern generators for locomotion, with special reference to vertebrates. *Ann Rev Neurosci* **8**: 233-261.

Heisenberg M (1994) Voluntariness (Willkurfahigkeit) and the general organization of behavior. In, *Flexibility and constraint in behavioral systems.* Greenspan RJ, and Kyriacou CP, eds. John Wiley & Sons: Chichester.

Heisenberg M, and Wolf R (1984) *Vision in Drosophila: Genetics of microbehavior.* Springer: Berlin.

Heisenberg M, and Wolf R (1988) Reafferent control of optomotor yaw torque in *Drosophila melanogaster. J Comp Physiol A* **163**:373-388.

Horridge GA (1989) Primitive vision based on sensing change. In, *Neurobiology of Sensory Systems.* Singh RN, and Strausfeld NJ, eds. Plenum Press: New York.

Hoyle G (1976) Arthropod walking. In, *Neural Control of Locomotion.* R Herman, S Grillner, PSG Stein, and DG Stuart, eds. Plenum Press: New York.

Jacklet JW (1989) *Neuronal and cellular oscillators.* Marcel Dekker: New York.

Kennedy JS (1986) Some current issues in orientation to odour sources. In, *Mechanisms in insect olfaction.* Payne TL, Birch MC, and Kennedy CEJ, eds. Clarendon Press: Oxford.

Kristan WB Jr (1980) Generation of rhythmic motor patterns. In, *Information Processing in the Nervous System.* HM Pinsker and WD Willis, eds. Raven Press: New York.

Lehrer M (1991) Bees which turn back and look. *Naturwissenschaften* **78**:274-276.

Lehrer M (1993) Why do bees turn back and look? *J Comp Physiol A* **172**:549-563.

Liddell EGT, and Sherrington CS (1924) Reflexes in response to stretch. *Proc Roy Soc* **96B**: 212-242.

Ludlow AR (1984) Application of computer modeling to behavioral coordination. PhD thesis. University of London.

Maes P (1991) *Designing Autonomous Agents.* MIT Press: Cambridge.

Möhl B (1988) Short-term learning during flight control in *Locusta migratoria. J Comp Physiol A* **156**:93-101.

Mura F, and Franceschini N (1994) Visual control of altitude and speed in a flying agent. In, *From animals to animats 3.* Cliff D, Husbands P, Meyer J-A, and Wilson SW, eds. MIT Press: Cambridge.

Olsen JF, and Suga N (1991a) Combination-sensitive neurons in the medial geniculate body of the mustached bat: Encoding of relative velocity information. *J Neurophysiol* **65**: 1254-1274.

Olsen JF, and Suga N (1991b) Combination-sensitive neurons in the medial geniculate body of the mustached bat: Encoding of target range information. *J Neurophysiol* **65**: 1275-1296.

Pearson KG (1972) Central programming and reflex control of walking in the cockroach. *J Exp Biol* **56**: 173-193.

Rheinlaender J, and Blatgen G (1982) The precision of auditory lateralization in the cricket, *Gryllus bimaculatus. Physiol Entomol* **7**:209-218.

Rheinlaender J, Gerhardt HC, Yager DD, and Capranica RR (1979) Accuracy of phonotaxis by the green treefrog (*Hyla cinerea*). *J Comp Physiol A* **133**:247-255.

Roberts A, and Roberts BL (1985) *Neural Origin of Rhythmic Movements.* Cambridge University Press: Cambridge.

Sandini G, Gandolfo F, Grosso E, Tistarelli M (1993) Vision during action. In, *Active perception.* Aloimonos Y, ed. Lawrence Erlbaum Associates: Hillsdale.

Schöne H (1984) *Spatial Orientation.* Princeton University Press: New Jersey.

Schmidt BC, and Ache BW (1979) Olfaction: response enhancement by flicking in a decapod crustacean. *Science* **205**: 204-206.

Schwenk K (1994) Why snakes have forked tongues. *Science* **263**: 1573-1577.

Selverston AI, and Moulins M (1985) Oscillatory neural networks. *Ann Rev Physiol* **47**: 29-48.

Sherrington CS (1906) *The Integrative Action of the Nervous System.* Raven Press: New York.

Sobel EC (1990) The locust's use of motion parallax to measure distance. *J Comp Physiol A* **167**: 579-588.

Willis MA, and Arbas EA (1996) Locomotory performance of moths flying upwind in a pheromone plume: individual variability in pheromone-modulated flight. Submitted.

Wilson DM (1961) The central nervous control of flight in a locust. *J Exp Biol* **38**: 471-490.

Wolf H, and Pearson KG (1987) Comparison of motor patterns in the intact and deafferented flight system of the locust. II. Intracellular recordings from flight motorneurons. *J Comp Physiol A* **160**: 269-279.

Wolf R, Voss A, Hein S, and Heisenberg M (1992) Can a fly ride a bicycle? *Phil Trans R Soc Lond B* **337**:261-269.

Zeil J (1993a) Orientation flights of solitary wasps (*Cerceris*; Sphecidae; Hymenoptera) I. Description of flight. *J Comp Physiol A* **172**:189-205.

Zeil J (1993b) Orientation flights of solitary wasps (*Cerceris*; Sphecidae; Hymenoptera) II. Similarities between orientation and return flights and the use of motion parallax. *J Comp Physiol A* **172**: 207-222.

Zill SN (1986) A model of pattern generation in cockroach walking reconsidered. *J Neurobiol* **17**: 317-328.

Zill SN, and Moran DT (1981) The exoskeleton and insect proprioception. III. Activity of tribal campaniform sensilla during walking in the american cockroach, *Periplaneta american*. *J Exp Biol* **94**:57-57.

Variable binding and predicate representation in a behavior-based architecture

Ian Horswill
The Institute for the Learning Sciences
Northwestern University
1890 Maple Avenue
Evanston, IL 60201
ian@ils.nwu.edu
Phone: (847) 467-1256
FAX: (847) 491-5258

May 9, 1996

Abstract

We present a technique called "role passing" for implementing bound variables and parameter passing in behavior-based systems. While intentionally limited, the system is very efficient, is very easily interfaced to perceptual and motor control systems, does not require the translation or fusion of sensory representations into a uniform interlingua, and is implementable in feed-forward parallel hardware. We present two implementations of role passing: a simple natural language system that answers questions using a real-time implementation of current theories of biological visual attention, and a robot that can approach, grab, carry, and/or deliver designated objects to designated locations.

1 Background

Much of the debate between classical and behavior-based architectures has centered on the issue of time-scale. A major driving force behind the popularity of hybrid symbolic/reactive architectures has been the need to simultaneously support both rapid response to a changing environment and means/ends reasoning. Hybrid architectures are often explicitly justified with claims that a reactive architecture is needed to support activity at one time scale (e.g. tenths of seconds) while high level reasoning is needed for longer time scales (e.g. tens of seconds) [2]. Were this true then a hundred-fold speed up in CPU power (which is not unreasonable) would be sufficient to allow the symbolic reasoning system to run on the tenths of seconds time-scale, and thus eliminate the need for the reactive system.

But surely this isn't true. The problems with reasoning systems do not have to do with the constant-factor differences between time scales but rather with combinatorics and with the problem of keeping symbolic world models up to date in a changing world. It is this latter epistemological problem on which I will focus in this paper.

I would argue that the success of reactive and behavior-based architectures is due to the fact that they have, so to speak, a tractable epistemology: representations are wires (or flip-flops) connected to sensory systems and are therefore continuously updated. When the world changes, the wire changes with it. The disadvantage of behavior-based systems is that wires are quite an impoverished representational medium. In particular, it is notoriously difficult to implement standard programming language features such as variable binding and dynamic trees. Attempts to incorporate symbolic processing into behavior-based systems have thus been limited to representations based on propositional logic (logic without variables or term expressions) rather than predicate logic, forcing the designer to use duplicate behaviors for each possible set of predicate arguments (see for example [10]).

Contrariwise, I would argue that the success of symbolic systems is due in large part to the richness of the representations they afford. Variables are just *useful*. Their problems lie in the fact that systems like STRIPS planners presuppose an accurate database of propositions that is more or less transparently kept up to date. This can be hard since perceptual systems

have internal resource limitations and so attentional resources need to be intelligently allocated based on knowledge of what the planner needs to know now or will need to know in the future. The worry here is that keeping the database up to date might be a harder planning problem than the original one.

Finally, I would argue that hybrid systems have been successful because, in practice, they don't implement the classical database epistemology. Most systems require that programmers hand-craft their code to know which aspects of the model to update when. Most hybrid systems don't implement a predicate planner at all, but only a graph-search engine that finds paths through a fixed map (see [5]). Nonetheless, these systems are effective because they give the programmer a rich representational language.

In this paper, I present a technique called "role-passing" that allows restricted forms variable binding and predicate representation, but is easily implemented in feed-forward parallel networks. The technique can be viewed as a generalization of Agre and Chapman's [1] indexical-functional, or "deictic" representation.

The connectionist literature has extensively studied the problem of efficiently implementing variables in fixed networks, as has the computer architecture community [9]. The problem is extremely difficult and involves an inherent trade-off between latency and hardware complexity. Tensor-product variable binding [14], is a sort of generalized adjacency matrix that can represent arbitrary trees, but at high cost. Touretzky [15] showed how to emulate a standard lisp engine in a Boltzmann network with both its advantages and disadvantages. Temporal-synchrony variable binding [13] allows small numbers of small dynamic structures to be manipulated through a time-domain multiplexing technique. It presupposes phased-locked-loop circuitry whose neurological plausibility is controversial. It also limited to implementing approximately ten bindings. Nevertheless, Henderson [4] has shown that TSVB is sufficient to implement a sophisticated description-based parser. Role-passing is most similar to TSVB in that it attempts only a partial solution of the binding problem. Its advantage is that, like traditional reactive systems, it has a clear and simple epistemology.

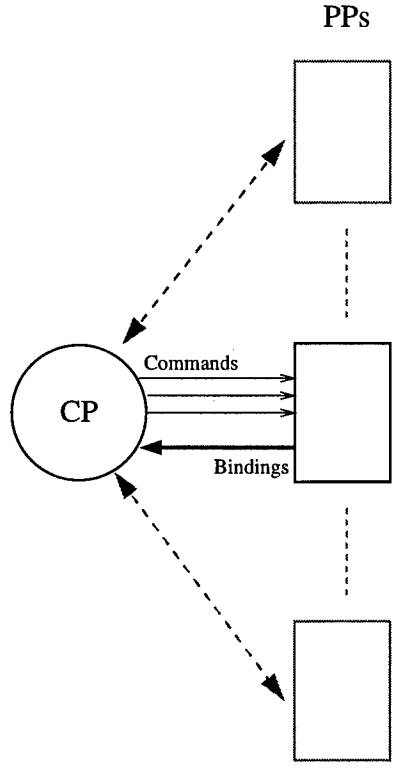


Figure 1: The role-passing architecture. PP command lines contain an activation (enable) signal and one or more role busses to specify the objects to be used as arguments.

2 A simplified cognitive architecture

We will divide the agent's control systems into a homogeneous *central processor* (CP) and a set of semi-autonomous *peripheral processors* (PPs). The CP implements a set of frames in the sense of the society of mind [11]: frames are fixed agencies that can be ei-

ther active or inactive. They are not data structures that are dynamically cons'ed and GC'ed.

Frame roles, such as OBJECT, DESTINATION, ACTOR, etc., can be filled either by the PPs (see below) or by other frames. In the latter case, the reference relations between the frames are implicit rather than being represented by pointers. Thus if INTEND and EAT are activated, EAT is understood to fill the ACTION slot of INTEND because (1) no other action is activated and (2) it doesn't make any sense for INTEND to fill a slot of EAT. If the possible tree structures formed by the frames satisfy an appropriate context-free grammar, it can be shown that (1) the trees can be uniquely represented by the activations of the frames, that is they can be reconstructed from a list of their vertices, (2) the activations can be consistency-checked by a shallow feed-forward network to verify that that the edge relationships of the tree are unambiguous, and (3) arbitrary sized trees can be unified in constant parallel time by performing a bitwise-or of their activations.

All other slots are bound in the PPs. PPs are standard behavior-based systems that implement modality-specific operations such as visual tracking, navigation, etc. Behaviors within a PP can be activated by the CP, by other behaviors in the PP or by other PPs. Unlike the CP, which uses a homogeneous propositional representation, PPs may use arbitrary specialized representations. However, PPs must satisfy a simple interface contract called *role-passing* (see figure 1). To understand role-passing, it is useful to distinguish *object* representations from representations of the situation as a whole. A visual tracker tracks a specific object. A sonar simply reports the distance to the nearest *stuff* without individuating the stuff into objects. Role-passing requires that:

- Object representations in PPs be tagged with one or more semantic role names drawn from a fixed set. That representation and role are then *bound* to one another.
- Representations bound to the same role must refer to the same object.
- When behaviors take object representations as arguments, they are specified by role name (e.g. `DESTINATION`) rather than by some name internal to the PP (e.g. `TRACKER-5`).
- Every PP reports the set of roles it binds to the CP.

Roles in role-passing are thus very much like symbols in a conventional S-expression oriented system, with three exceptions:

- Roles have indexical denotation. Binding a role to an object is an attentive operation and that binding can be changed at any time.
- Only one role with a given name can be represented at a time. Two frames with `DESTINATION` slots cannot be co-activated unless they share the same destination.
- Information about the object filling a role is not stored in a homogeneous database, but is distributed throughout the PPs. A given object can have different representations in different sensory modalities. The set of modalities in which it is represented can change over time.

2.1 Implementing role-passing

Role-passing is easily implemented in parallel distributed hardware. The set of possible roles is fixed in advance. A role or set of roles can then be represented as a bit-vector, one bit per role. Object representations are tagged with roles by adding a latch to them to hold the bit-set of roles to which it is bound.

Each PP contains one or more pools of object representations. Representations within a pool are allocated on a least-recently-used basis. A *port* is a connection between a pool and some other part of the agent. It consists of a role bus and a data bus. When the role bus is driven with a set of roles, comparators compare them to the latched roles of each object representation in the pool. The matching representation drives the data bus with its data. The resulting circuitry is nearly identical to a standard LRU cache (see figure 2).

For example, suppose we have three PPs: a visual tracking system that contains two trackers within it, an approach-and-follow system that can perform visual servoing to approach an arbitrary tracked object, and a manipulation system that can perform visual servoing to pick up an arbitrary tracked object. The servoing systems are connected to the trackers by ports that allow them to get the coordinates of the centroid of a specified tracker. Suppose one tracker was targeted to a person and bound to the `DESTINATION` role while the other tracker was targeted to a box and bound to the `OBJECT` role. The CP can then direct the robot to bring the box to the person using the following procedure:

- Drive the role bus of the port connecting the follower to the tracker with `OBJECT`, and enable the follower. The robot approaches the box.
- When the follower signals completion, disable it, then drive the manipulation systems role bus with `OBJECT` and enable it. The robot picks up the box.
- When the manipulation system signals completion, drive the following system's role bus with

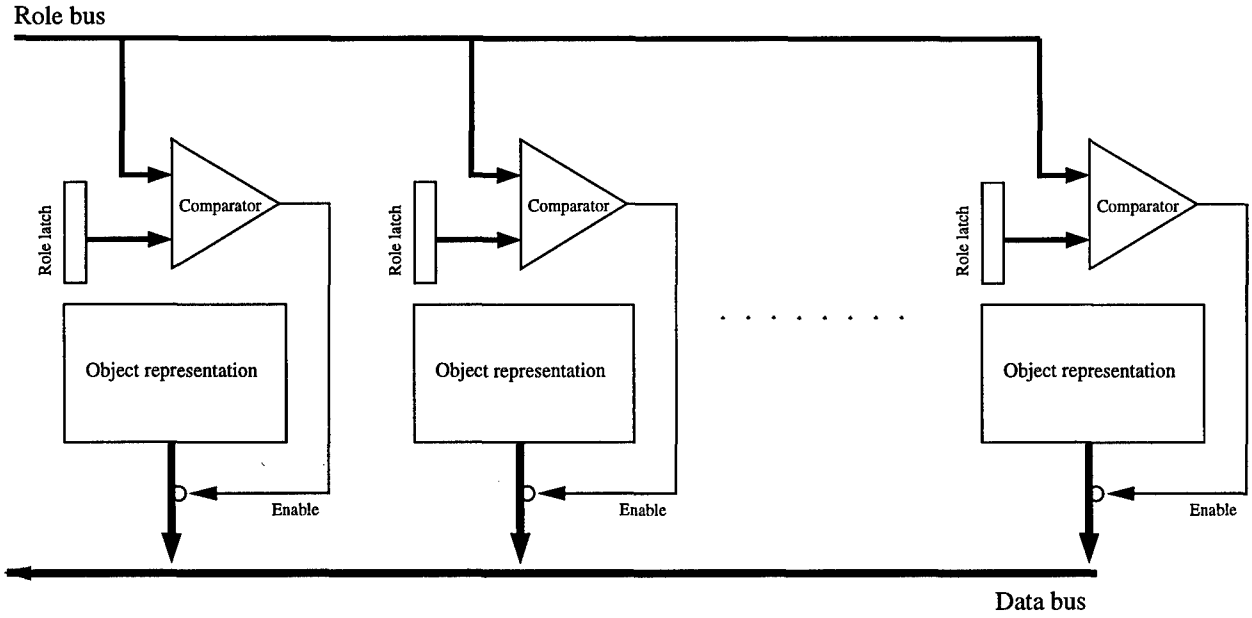


Figure 2: Port implementation. A given pool of object representations may have many access ports.

DESTINATION and enable it. The robot approaches the person.

- Disable the manipulation system, causing it to drop the contents of the hand.

2.2 *Automatic type conversion*

A single object can be represented in many different modalities. However, one often wants it to be represented in a particular modality. For example, the linguistic subsystem (see below) may have a description of an object, but the GET behavior may need a visual marker (i.e. tracker) for its target. A description may be converted to a marker by invoking the visual search system (again, see below), but it needs to be told to do so, and it needs to be told what role to convert. We would like this to happen automatically, so that GET need only drive a wire asserting that it wants its TARGET role bound to a marker.

The architecture already keeps a table (i.e. a bus of wires) listing what roles are bound in what modalities. By keeping an additional table of what roles are needed in what modalities, we can provide support for automatic type conversion (see figure 3). To automatically convert descriptions to markers, we add circuitry to AND the set of roles bound in the description buffer to the set of roles needed in visual markers, and drive the resulting roles on the role bus of the visual search system. Now when the GET agency is activated, it can simply assert the need-target-in-marker wire and wait until the target-in-marker wire is asserted.

Type conversion allows a limited form of automatic subgoaling. Rather than using an explicit sequencer to control the PPs for the carrying task discussed above, we could implement most of the work with type conversion. In fact, we can handle the task even if we only have descriptions of the object and destination, rather than visual bindings:

- The DELIVER agency simply signals that it wants OBJECT bound by *the hand.*
- The visual search system is activated to convert the description form of OBJECT into a visual marker.
- GET is now activated to convert the marker into a hand binding. It activates the following system to get within range.
- Once the object is picked up, it is listed by the manipulation PP as bound in the hand, allowing TAKE to continue.
- DELIVER now invokes the follower on DESTINATION.

3 Ludwig: natural language in a distributed system

Ludwig is a minimal implementation of the role-passing scheme. It contains three PPs: a *description buffer* that holds predigested noun-phrases, a real-time implementation of Ullman's visual routine processor (VRP) theory [16][3], and a simple motor-control system.

The VRP is able to dynamically direct attention

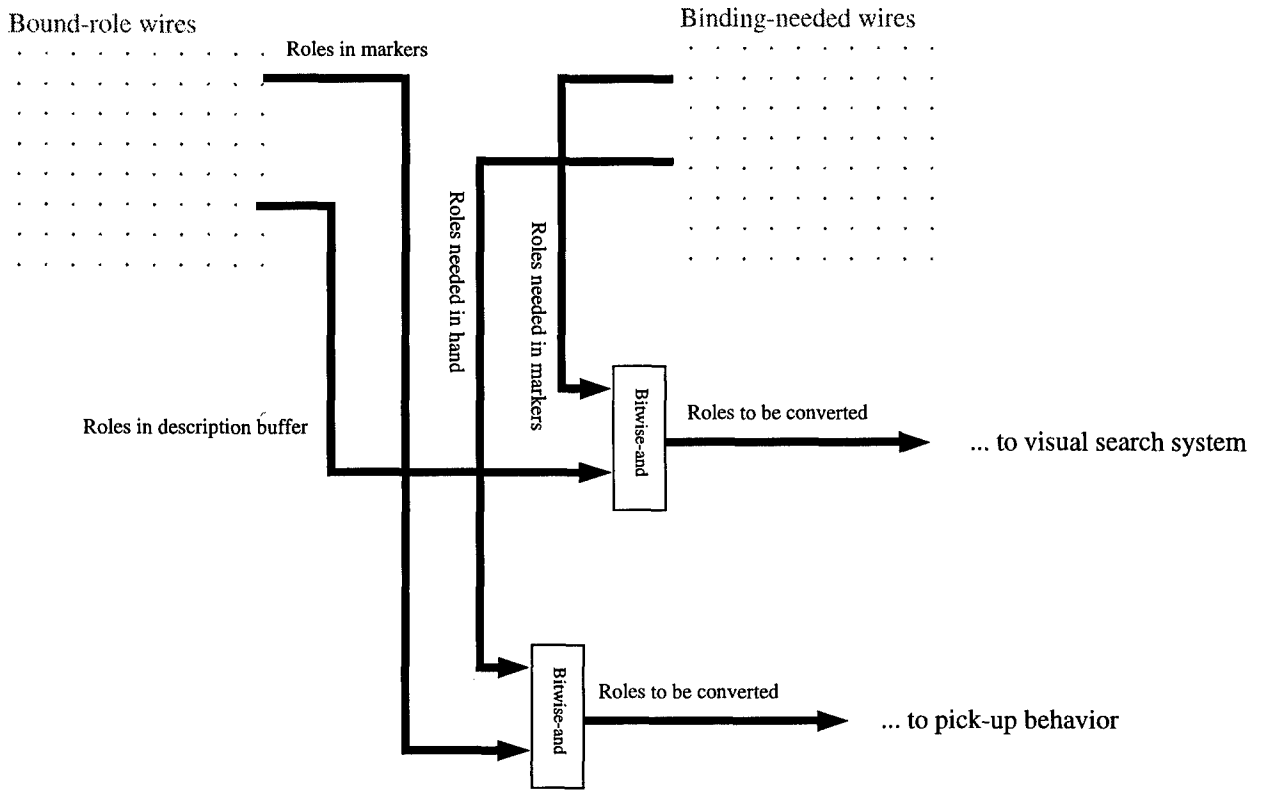


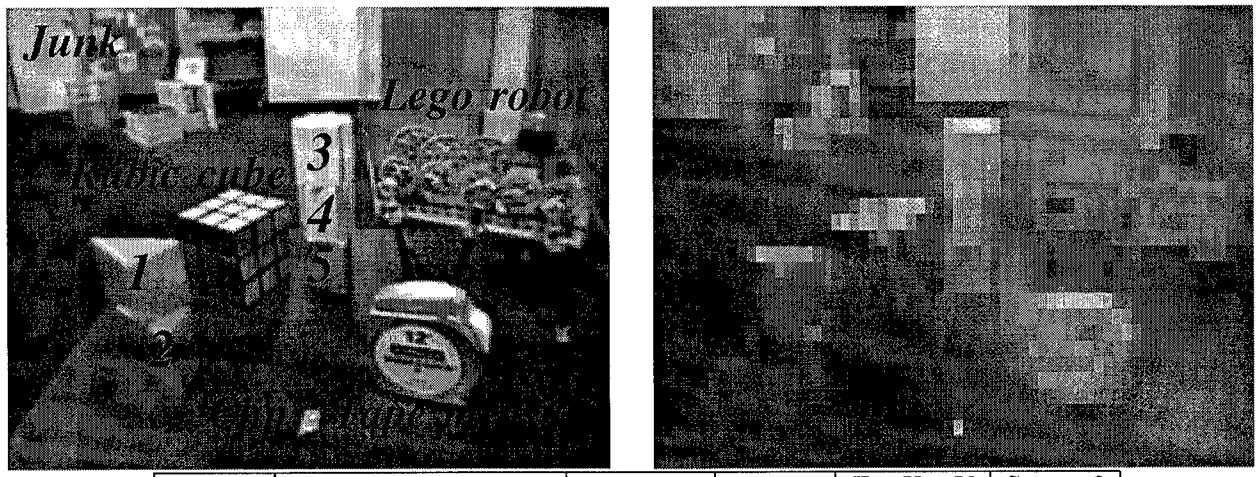
Figure 3: Implementing automatic type conversion in parallel hardware.

to image regions with specified properties or spatial relations to other image regions. The description buffer stores predigested noun phrases as shift-registers containing sequences of primitive VRP instructions. When a description buffer entry corresponding to the noun phrase "blue block on a red block" is passed to the VRP, the VRP runs the instructions and binds a visual marker to a blue block that is on a red block. From an abstract standpoint, the VRP behaves like a Prolog engine that represents variables as image-plane markers rather than symbolic expressions. It can backtrack and solve arbitrary Horn clauses, subject to certain resource limitations. See [7] for a full discussion of the vision system, its control hardware, and its theoretical limitations.

Ludwig's vision system is written in C and runs at 5 frames per second on low-cost hardware. The C code simulates a set of parallel processes that compute a set of low-level visual maps, an attention system based on current biological attention theories [8], and a set of visual markers for recording image locations extracted by the attention system (see [7]). The rest of the system is implemented directly in (simulated) digital logic: gates, registers, busses, etc.

The system is extremely parallel. Parsing, semantic analysis, and visual search execute concurrently in separate pipelines. Its language system translates successive words into VRP instructions and stores them in the current description buffer entry. A set of finite-state machines detect phrase boundaries and switch to the next description buffer entry. Another FSM monitors the current verb and tags description buffer entries with the appropriate role names (`THEME` and `PREDICATE`). Other circuitry implements the automatic conversion of `THEME` descriptions into markers.

The processing of a query such as "is the block on the red block green?" goes like this. The words are presented to the system one at a time and are shifted into the description buffer. The description buffer is filled with the visual translation of "block on red block" and that buffer entry is bound to `THEME`. As soon as the `THEME` description is complete, it is automatically converted into a visual marker by the visual search system. In parallel, the language system continues to accept words, this time placing "green" in a new description buffer entry and binding it to `PREDICATE`. Another state machine now detects that `THEME` is bound in the visual domain and `PREDICATE` in the description domain and invokes visual search on the predicate, testing whether the theme satisfies



Word	Parser	Instruction	VRP	T	X	Y	Success?
tell	V						
me	N NP Experiencer						
if							
a							
blue		blue(T)					
block	N NP Theme						
is	V		blue(T)	1			Yes
Theme object found (block 1); VRP waits for rest of sentence							
on	P	on(T,X)					
another							
block	N						
on	P	on(X,Y)					
another							
block	N NP Predicate						
stop			on(T,X)	1	2		Yes
			on(X,Y)	1	2		No
Theme failed predicate test							
			blue(T)	3	2		Yes
Found new theme (block 3)							
			on(T,X)	3	4		Yes
			on(X,Y)	3	4	5	Yes
Theme passed predicate							
Ludwig answers "yes."							

Figure 4: Example scene and time course of the query "tell me if a blue block is on another block on another block." Note the parser and the VRP run in parallel. "Parser" gives the internal state of the parser FSMs. "Instruction" is the current instruction being written to the description buffer. "VRP" is the current instruction being fetched by the VRP *from* the description buffer. "T," "X," and "Y" give the bindings of theme marker and two scratch markers to the numbered blocks in the image. "Succeed?" indicates whether the VRP operation succeeded or whether the VRP has to backtrack. Blocks 1 and 3 are blue (by Ludwig's definition of blue). The left image is full resolution and has been contrast-enhanced to aid viewing. The right image shows the resolution at which the vision system runs.

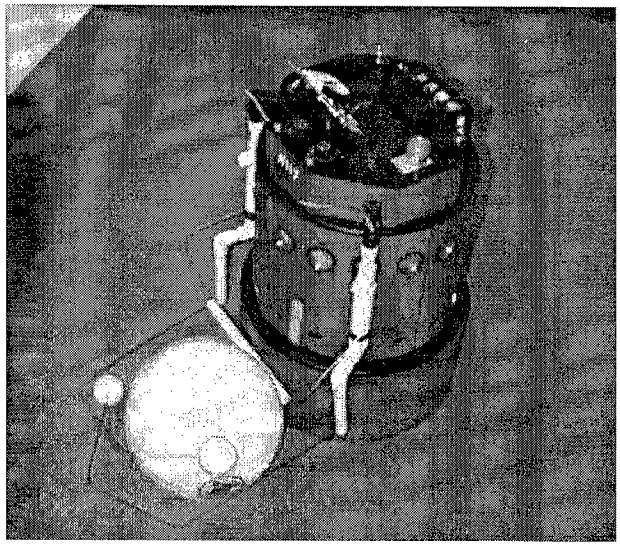


Figure 5: Kludge the robot grasping a ball.

the predicate. Depending on the result, the system replies "yes," "no," or determines that it had bound the theme to the wrong block and tries an alternate choice.

4 Delivering balls

Recently, we have developed a more complete example of role passing at Northwestern University. Our system implements the DELIVER, GET, and follow operations discussed above. It consists of a commercial robot base (RWI B14S) with an MIT Cheap Vision Machine and a set of homemade mandibles (1dof arms) (see figure 5).

The vision system implements adaptive tracking of multiple homogeneous colored objects and visual collision avoidance. Trackers can visually acquire, track, and report the position of objects given their descriptions. At present, the visual system only implements single-predicate descriptions (colors), but we intend to integrate Ludwig's search capabilities into the tracking system soon. The mandibles can be used to grab, carry, or deposit plastic children's balls. Note that, unlike the Ludwig system, the ball delivery system does not simulate the network down to the gate level. Ports and pools are implemented as lisp code that acts as would the hardware implementation, but which runs faster on a serial machine. From the standpoint of data flow, the system is a feed-forward dependency network that flows from the visual trackers and other sensors, through the control system, and into the effectors.

The delivery system uses four PPs: a description buffer as in ludwig, the tracking system which plays a

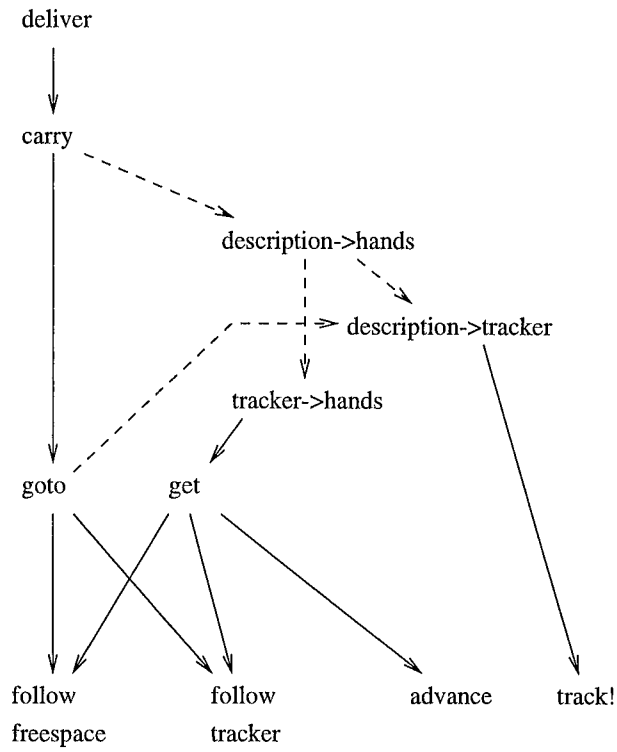


Figure 6: Control hierarchy of the carry/deliver system. Activation flows top to bottom. Solid arrows denote direct explicit activation. Dashed arrows denote indirect activation through type conversion.

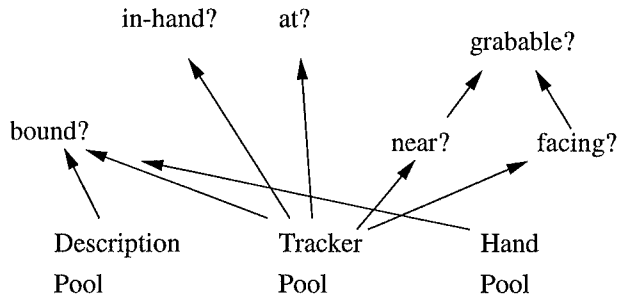


Figure 7: Data hierarchy of the carry/deliver system. Activation flows bottom to top. Each predicate is represented as a bus of wires giving truth values for each role. Logical connectives are implemented by bitwise and/or/not.

role analogous to the markers in Ludwig, a collision avoidance system [6], and the mandibles. The description buffer and tracking system implement pools of bindable representations. The hand pool is a degenerate case in that our current system can only carry one object. We hope to add the tracking of odometric coordinates for fixed points as another pool in the next few weeks. The CP implements a hierarchy of control agencies (see figure 6) and a set of busses (bit-vectors) representing predicates (see figure 7). Each predicate has a bus and each role has a wire in each bus. A wire is active iff its predicate is true of its role.

On each clock tick, the PPs re-sample the perceptual data. In particular, the tracker PP recomputes the low level predicate of each tracker and drives the predicate busses accordingly. Compound predicates are then computed as bitwise ands and ors of the busses (e.g. grabable(x) = near(x) and facing(x)).

The system implements automatic conversion of descriptions to trackers to hands. As with Ludwig, descriptions are converted to trackers on demand by visual search. Trackers are "converted" to hands by activating the GET unit. Descriptions are converted to hands by chaining these conversions.

For example, the robot can be instructed to deliver a pink ball to a person wearing blue jeans by binding the normalized color coordinates of pink to a description with the OBJECT role, the normalized color coordinates of blue to a description with the DESTINATION role, and activating the DELIVER unit in the control hierarchy. DELIVER implements the rule:

```
(if (enabled? deliver)
    (and (need-object-in-hand OBJECT)
         (if (in-hand? OBJECT)
             (and (enable goto)
                  (if (at? DESTINATION)
                      (open-mandibles!))
                  ))))
```

When DELIVER is enabled, it drives the OBJECT wire on the need-in-hand bus, triggering a series of conversions until the object is bound in the hand. When the OBJECT wire of the in-hand bus is true, DELIVER drives the enable line of GOTO. When the DESTINATION wire of the at? bus is true, it sends a signal to the mandible controller to let go of the object.

GOTO takes its argument implicitly from the DESTINATION role. It implements the rule:

```
(if (enabled? goto)
    (and (need-object-in-tracker
           DESTINATION)
         (if (tracking? DESTINATION)
             (and (enable follow-freespace)
                  (enable follow-object)
                  (set-target
                    DESTINATION)))))
```

Follow-freespace and follow-object are primitives of the freespace-follower and tracker PPs respectively. Set-target drives the target port of the tracker PP with the DESTINATION role. Thus the tracker follows the DESTINATION when GOTO is active and the OBJECT when GET is active. A more elegant approach would be to have the follower always follow a single role, TARGET, and then dynamically equate TARGET to either DESTINATION or OBJECT at run time. For n roles and m trackers, this can be implemented in $O(\log n)$ time and $O(n^2m)$ hardware using a scheme similar to matrix multiplication, but we have not yet implemented this technique.

Note that in reality, the system is implemented as lisp code, so roles are represented as bits (OBJECT=1, SOURCE=2, DESTINATION=4, etc.) and sets of roles as integers (i.e. { SOURCE, OBJECT}=3). In-hand? is then defined as:

```
(define (in-hand? role)
  (not (zero? (bitwise-and role
                           in-hand-roles))))
```

so predicate applications and assertions are also fast on serial hardware.

A typical run of the system goes as follows. The system starts out with descriptions of DESTINATION and OBJECT, but cannot see them. Deliver asserts that it needs the OBJECT in the hand, so the conversion system initiates a visual search for the description bound by OBJECT. Once OBJECT is found, the conversion system activates GET, which drives OBJECT on the target port, and the robot drives to the object. When grabable?(OBJECT) becomes true, it opens the arms and approaches it until the visual system asserts in-hand?(OBJECT), at which point the hand system binds OBJECT, deactivating the conversion system and thus GET. Deliver is now able to activate GOTO, which asserts that it needs DESTINATION bound by a tracker, which reactivates the conversion system, but this time on the DESTINATION role. When the robot visually acquires the DESTINATION, GOTO drives DESTINATION on the target port and the robot drives to the destination. If along the way we intervene and remove the ball from the robot's grasp, OBJECT becomes unbound in the hand because the hand sys-

tem continuously monitors the trackers. The following then happens immediately on the next clock tick:

- GOTO is disabled,
- The conversion system is reactivated on OBJECT,
- GET is reactivated (assuming the OBJECT is still in view),
- FOLLOW-OBJECT remains active, but the target port is driven with OBJECT, so the robot turns to the ball.

When robot has finished regrabbing the ball, GOTO is reenabled, and the robot continues on its way. Eventually, the visual system asserts at?(DESTINATION) and the robot releases the ball and stops.

5 Discussion

Role passing is an attempt to import as much of the expressiveness and generativity of symbolic architectures as possible into behavior-based architectures while preserving the latter's simple and clear interface to perception and motor control. Unlike previous behavior-based systems (e.g. [10]), the same grab unit can grab many different objects depending on what bindings are in effect. Because real agents have many disparate sensory modalities, sensory representations of objects will inevitably be distributed across many different representations. Rather than forcing the translation of all data into a uniform interlingua, role passing transparently converts between representations on demand, when possible.

The Ludwig system, as a natural language system, is syntactically and semantically crude. Its purpose is to give an example of how many features of classical problem solvers can be efficiently simulated without needing to fuse all sensory information into a uniform database. Rather than implementing a uniform *representational* medium [12] for objects, Ludwig implements a uniform *means of access* to object representations.

The fetch and deliver system is also simple, but implements a useful task in a manner that we hope will be generative. Given this system as a substrate, we can add new types of PPs to track other kinds of objects or to track objects out of view using odometry and modify the lowest level follower routine to accept tracking data from the new pools. Then higher level agents like DELIVER can use objects bound in these domains without even knowing where they're bound. Similarly, other agencies can be designed to leverage off of the capabilities of the original agencies by activating them and passing objects to them as parameters in the same way GET is called.

The role-passing system is work in progress. It remains to be seen how far it can be scaled. For example, it is not likely to support theorem proving to minimaxing tree search. Nevertheless, it demonstrates that many useful features of classical AI systems can be safely imported into behavior-based systems without sacrificing reactivity.

Acknowledgments

This paper describes research performed in part at the MIT Artificial Intelligence Laboratory and at The Institute for the Learning Sciences at Northwestern University. Many thanks are due to Chris Barnhart for building the Cheap Vision Machine, to Pete Beim and Clifton Poole for designing and building the mandibles on Kluge, and Pete Beim and Ivan Yen for earlier work on the grabbing and carrying tasks. Support for this research was provided in part by the University Research Initiative under Office of Naval Research contract N00014–86–K–0685, and in part by the Advanced Research Projects Agency under Office of Naval Research contract N00014–85–K–0124, and in part by the National Science Foundation under grant number IRI–9407273. The Institute for the Learning Sciences was established in 1989 with the support of Anderson Consulting, part of the Arthur Anderson Worldwide Organization.

References

[1] Philip E. Agre and David Chapman. Pengi: An implementation of a theory of activity. In *Proceedings of the Sixth National Conference on Artificial Intelligence*, pages 268–272, 1987.

[2] R. Peter Bonasso, R. James Firby, Erann Gat, David Kortenkamp, David P. Miller, and Marc G. Slack. Experiences with an architecture for intelligent reactive agents. *Journal of Theoretical and Experimental AI.* In press.

[3] David Chapman. Vision, instruction, and action. Technical Report 1204, Massachusetts Institute of Technology, Artificial Intelligence Lab, April 1990.

[4] James Brinton Henderson. *Description-Based Parsing in a Connectionist Network.* PhD thesis, University of Pennsylvania, 1994.

[5] Henry Hexmoor, Ian Horswill, and David Kortenkamp, editors. *Special issue on software architectures for physical agents, Journal of Theoretical and Experimental AI.* In press.

[6] Ian Horswill. Polly: A vision-based artificial agent. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pages 824–829. AAAI, MIT Press, 1993.

[7] Ian Horswill. Visual routines and visual search. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, Montreal, August 1995.

[8] C. Koch and S. Ullman. Shifts in selective visual attention: Towards the underlying neural circuitry. *Human Neurobiology*, 2:219–227, 1985.

[9] F. Thomson Leighton. *Introduction to Parallel Algorithms and Architectures.* Morgan Kaufman, San Mateo, CA, 1992.

[10] Pattie Maes. How to do the right thing. AI Memo 1180, MIT Artificial Intelligence Laboratory, December 1989.

[11] Marvin Minsky. *The Society of Mind.* Simon and Schuster, New York, NY, 1986.

[12] Allen Newell. *Unified Theories of Cognition.* Harvard University Press, Cambridge, MA, 1990.

[13] Lokendra Shastri and Venkat Ajjanagadde. From simple associations to systematic reasoning: A connectionist representation of rules, variables, and dynamic bindings using temporal synchrony. *Behavioral and Brain Sciences*, 16, 1993.

[14] Paul Smolensky. Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artificial Intelligence*, 46(1–2):159–216, 1990.

[15] D. S. Touretzky. Boltzcons: Reconciling connectionism with the recursive nature of stacks and trees. *Artificial Intelligence*, 46(1–2), 1990.

[16] Shimon Ullman. Visual routines. *Cognition*, 18:97–159, 1984.

THE EXPERIMENTAL STUDY AND COMPUTER SIMULATION OF FISH BEHAVIOR IN THE UNIFORM ENVIRONMENT

V.A. NEPOMNYASHCHIKH & Vera A. GREMYATCHIKH
Institute of Biology of Inland Waters, Russian Academy of Sciences,
152742 Borok, Yaroslavl obl., Russia (nepom@ibiw.yaroslavl.su)

Abstract

We have studied experimentally the carp exploratory behavior in the circular corridor under uniform illumination and in the absence of external landmarks. The behavior was not uniform despite the uniformity of environment. The alternation of two modes of behavior, search (accompanied by high turn frequency) and ranging (with few or no turns), was observed. The existence of distinct modes manifested itself in the positive correlation between turn frequencies at successive time intervals, as well as in abrupt switching between high and low turn frequencies. We developed a model based on the one-dimension map: $X_n = \lambda_n X_{n-1} (1-X_{n-1})$, where X is the tendency to turn, and parameter λ is influenced by Gaussian white noise caused by spontaneous nervous activity. There was a correlation between successive values of the tendency, as well as switches from series of low values to series of high ones under influence of the noise. The behavior of the model is the case of noise-induced phase transitions and matches experimental data qualitatively. It is concluded that a noise generated by the nervous system can play a role in the shaping of animal behavior.

1. Introduction

The exploratory and searching behavior in animals consists generally of restricted search within certain areas and ranging between these areas. The restricted search is usually a response to a stimulation. In fishes, for instance, the discovery of a food causes a decrease in the swimming velocity as well as frequent changes of the direction of the swimming. As a result, a fish tends to stay within restricted area. If no more food is discovered, a fish changes the direction more rarely and leaves the area (Thomas, 1974; Hunter & Thomas, 1974; Munk & Kiorboe, 1985; Coughlin et al., 1992). The similar response is also caused by various visual landmarks like water plants which are signals of the presence of food for carp (Scott & Crossman, 1973). In laboratory, visual stimuli evoke the restricted search in goldfish (Warburton, 1990), carp (Gdovskii et al., 1994) and tilapia (Nepomnyashchikh & Gremyatchikh, 1993).

The same dependence of the restricted search on external stimulation had been observed in various animal groups and is used as the basis for models of searching behavior, including klinokinesis (Fraenkel & Gunn, 1961; Benhamou & Bovet, 1989) and optimal foraging (Charnov, 1976; Stephens & Krebs, 1986.). The observed structure of the behavior - the alternation of restricted search and ranging - is derived in these models from a structure of an environment.

It is known, however, that the similar structure may be observed also in those environments, where no distinct 'areas' can be discriminated. Goldfish, for example, explore restricted areas where no landmarks or other objects may be used to discriminate between areas (Kleerekoper et al., 1974). The alternation of restricted search and ranging in an uniform environment was also found in tilapia (Nepomnyashchikh & Gremyatchikh, 1993) and carp (Nepomnyashchikh et al., 1995). These data suggest that an internal mechanism exists which generates the observed spontaneous alternation. We have attempted to evaluate the nature of this mechanism in the common carp Cyprinus carpio L. using laboratory experiments and the mathematical modeling.

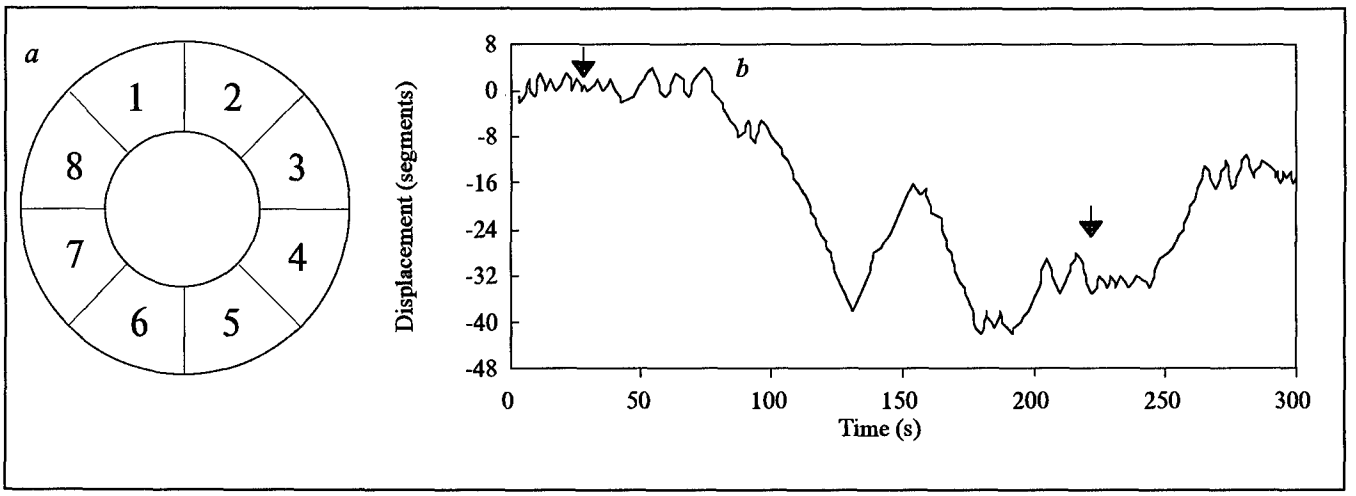


Fig. 1. (a) Diagram of the circular corridor. Segments are marked with numbers and divided by lines. (b) Path of a carp along 'infinite' wall. Positive numbers correspond to clockwise displacement from the start point (0). Perimeter of the corridor is equal to eight segments. Two series of turns are marked with arrows.

2. Materials and methods

2.1. *Fishes*

Carp had been collected in the wild and kept in an aquarium during four months prior to experiments at water temperature 19-20 °C. Fishes were fed with balanced artificial fish food and aged 7 to 8 month (body length 45-55 mm) at the time of experiments.

2.2. *Experimental Device*

We had attempted to make fish's swimming 'one-dimensional' in order to simplify the data analysis. To this end, a fish was placed into the corridor made of uniformly white plastic, circular in shape and provided with flat transparent glass cover (Fig. 1a). Fish can turn inside the corridor, but its motion was generally constrained within clockwise and counter clockwise directions along walls. The corridor was filled with water up to the cover in order to prevent air bubbles or specks on water surface to attract fishes. The only light source was the beam focused at the center of the circle to provide an uniform illumination in the corridor (40 lx on water surface). The outer diameter of corridor was 144, inner diameter - 89, and the depth - 55 mm. The device was installed on the massive concrete column to protect fishes from vibrations.

2.3. *Procedure*

Fishes were fed ad libitum the day before the experiment, and no more food had been provided until the observation finished. Individual fishes were placed in the corridor at water temperature 19-20 °C, and their motion was monitored for 30 min after 1 min adaptation. The b/w video camera was used for observation and TV screen was divided into 8 numbered sectors (Fig. 1a). When a fish entered a sector, experimenter pressed the respective button at PC keypad. The behavior of 40 individuals had been recorded, no individual being observed more than once.

2.4. *Data analysis*

Data were stored on the disk and then processed in order to count the frequency of turns per time unit, as well as the frequency per segment summed up for the period of observation. The turn was recorded if fish had returned to previous segment after visiting next one (for example, in the sequence 3-4-3 of segments visited the turn in segment 4 should be recorded). The fishes' behavior was not always a stationary process: the turn frequency tended to increase during first minutes of observation. In order to prevent the tendency to interfere with the analysis, we derived stationary fragments from each individual record. For this purpose, cumulative turn frequency against time was plotted and those fragments of records were only used for the further analysis which fitted satisfactorily a linear regression and were not shorter than 20 min. Fragments were also excluded from the analysis if a fish had stopped in the corridor.

In order to represent a record graphically, we 'unfolded' the fish's path in the corridor so as it looks like the swimming along an infinite wall (Fig 1b). We also divided individual records into equal time intervals 10 to 80 s long and calculated frequencies of turns in each interval (Fig 2). Correlation coefficient, Cc, of turn frequencies in successive intervals was then calculated for each fish separately.

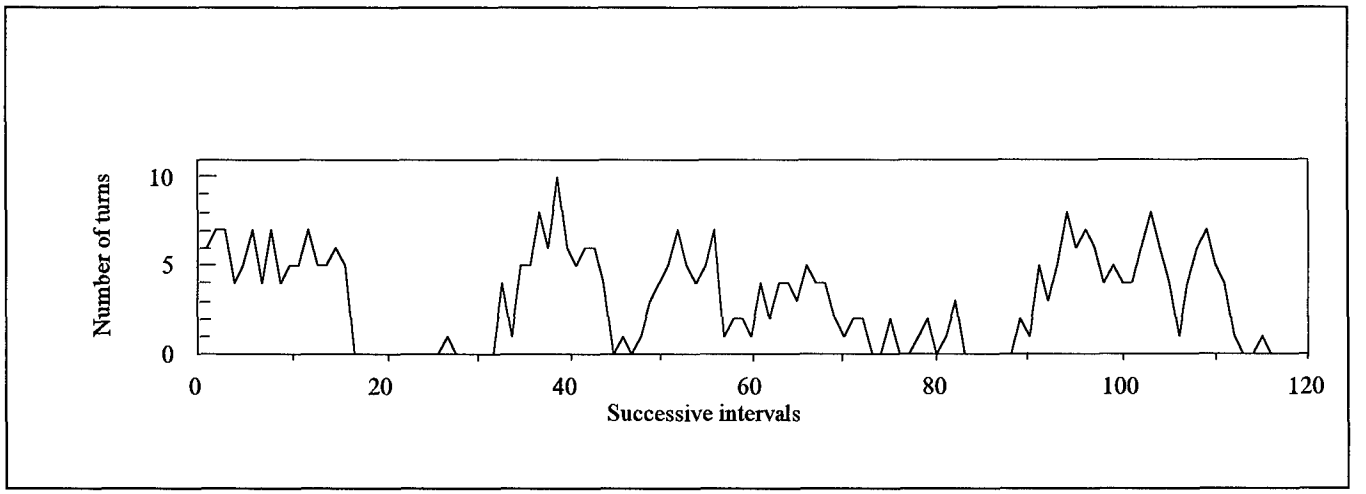


Fig. 2. Number of turns of individual carp in successive time intervals. Each interval is equal to 10 s.

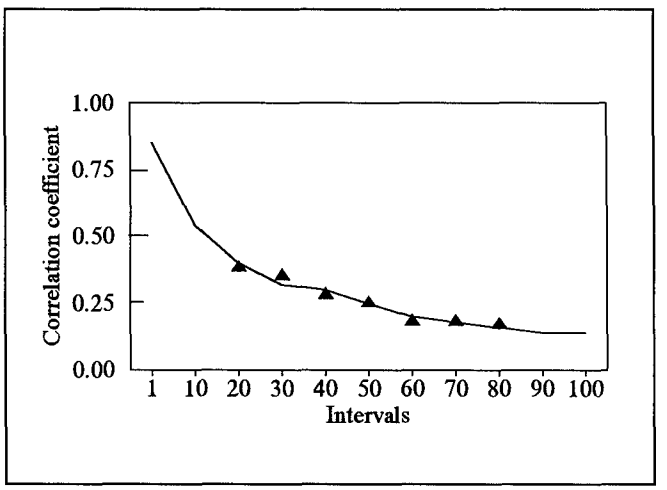


Fig. 3. Correlation coefficients between turn frequencies in carp (triangles) and between turn tendencies in the model (line). in time intervals of different lengths. Length of intervals is shown in seconds for carp and in steps for the model.

3. Results

Of 40 individual records we had obtained in the experiment, 28 had stationary fragments long enough to be analyzed. The alternation of two behavioral modes had been revealed in most records, as exemplified in Figs. 1b and 2, which show the behavior of carp in two different ways. It can be seen from both graphs that turns are grouped by series in time and space, while series are divided by runs with few or no turns. This grouping of turns had been proven true by calculating Ccs between turn frequencies. Parametric Ccs averaged for 28 individuals for intervals 20 to 80 s are shown in Fig. 3. All averages exceed zero significantly ($p<0.05$, t-test, two-tailed). Two-way ANOVA (fish x interval length) yielded no significant differences between carp individuals ($F=0.91$, $p>0.05$), but showed significant dependence of Ccs on the length of intervals ($F=37.41$, $p<0.0001$).

Nonparametric Spearman's Ccs were calculated for 10 s intervals in order to avoid errors, because most of these intervals contained few or no turns, and frequency distribution of turns was far from Gaussian. All 28 Ccs were positive (the average $Cc=0.41\pm0.07$, $p=0.05$), 27 Ccs being significant ($p<0.05$).

The series of turns resulted in keeping a fish within a restricted area along the 'infinite' wall (Fig. 1b). The frequency of turns in different segments of the corridor did not differ significantly from the uniform distribution neither in individual records, nor in summed-up data ($p>>0.05$, Chi-square test). We failed also to reveal any periodicity in turn frequencies using the auto-correlation analysis.

4. Model

4.1. *Control mechanism*

There is the evident orderliness in carp's behavior which consists in the correlation of turn frequencies and alternation of two modes of behavior. On the other hand, the behavior is not periodic. When modeling the behavior, it seems logical to look for a mechanism which may account for both features.

We assume that the observed turn frequency depends on an inner variable: the turn tendency, X. The frequency of turns per time interval in our experiment is the discrete variable and it is reasonable to let X change by discrete steps (n, n+1) in the model in order to make it possible to compare the results of simulation with experimental data. Also, the tendency must be kept within a restricted range by means of a feedback. We had chosen the well-studied one-dimension map (May, 1976) to fulfill these

assumptions:

$$X_n = \lambda_n X_{n-1} (1-X_{n-1}), \quad 0< X <1, \quad 0<\lambda<4, \quad (1)$$

where λ_n is the parameter to be defined below. The higher the X is at nth step, the more turns a fish performs within nth interval. For the sake of simplicity, we assume the turn frequency to be linearly proportional to turn tendency.

4.2. *Arousing input*

We assume also that the turn tendency depends on an arousing input from various nervous structures. This is the realistic assumption, based on numerous experimental data. Fishes' exploratory behavior is aroused by the forebrain's activity (Aronson, 1970) and suppressed in forebrain-ectomised fishes (Segaar, 1961), including goldfish (Janzen, 1933). The similar arousing effect exerts the olfactory system, because a chronic olfactory deprivation suppress an exploration (Atema et al., 1969). The restricted search caused by a visual stimulus in carp was also suppressed both by anosmy (Gdovskii et al, 1994) and bulbectomy (Gdovskii, personal communication). The value of the arousing input is represented by λ_n in our model.

4.3. *Noise*

The arousing input is not likely to be constant. We assume that there are numerous sources of the input, and the activity of any source varies absolutely independently of other ones. In this case, the summed-up variations of the input may be viewed upon as the Gaussian noise. We assume also that the noise is not correlated in time (white noise). Of course, both assumptions are tentative ones and were made in order to simplify the model. Basing on the assumptions, the parameter may be now defined:

$$\lambda_n = \mu + \sigma\xi_n, \quad (2)$$

where μ is the average value of the input, ξ_n is the Gaussian noise scaled in the range -0.5 to +0.5, and σ is the constant.

4.4. *Computational details*

Borland Pascal 7.0 program was run on an PC IBM for simulations. We computed the model's behavior using $X_0 = 0.1$ and various values of the average parameter ($1.0<\mu<1.5$) and noise constant ($0.0<\sigma<3.0$). Each run consisted of 12,000 steps and 2,000 last steps only were used for an analysis in order to achieve a stationary behavior. Pascal random number generator produced a sequence of values, which were then transformed in order to simulate Gaussian noise distribution with the standard deviation 2.0. Correlation coefficients, Ccs, between noise values at successive steps did not exceed ±0.05. The subsequent analysis was curried out in the way similar to the one used for experimental data. The records were divided into equal intervals 10 to 100 steps long, and values of X were averaged across an interval. Then parametric Ccs between successive steps, as well as between averages in successive intervals were calculated for 100 independent runs.

4.5. *Behavior of the model*

Averaged Ccs for successive values of X obtained at $\mu=1.151$ and $\sigma=2.3$ are shown in Fig. 3. As in the carp behavior, there is positive correlation in the behavior of the model. Both in carp and in the model, this correlation decreases as the length of intervals increases, but still take place within a rather wide range of intervals. We do not know which length of interval in fishes' behavior one step of model should be compared with, but the relative scale-independence of the correlation suggests that qualitative comparisons may be made using different time scales.

Changes of turn tendency at each step of simulation obtained under the same conditions are shown in Fig. 4a. Series of high and low values of the tendency are observed similar to series of high and low turn frequencies in fishes (see Fig. 2). Fig. 4b shows turn tendencies averaged across 10 step's intervals. The similar series take place in spite of the change of the scale. Simulations using intervals up to 50 steps resulted in qualitatively the same pattern of changes in the turn tendency. These results are in line with our suggestion: fishes' and model's behavior may be compared at different scales.

No signs of periodicity were revealed using the auto-correlation analysis. These data show that there is both the orderliness and non-periodical changes in the model's behavior. We will see in a moment how these features are generated in the model. When the parameter λ_n is kept constant ($\lambda_n=\mu$), the behavior depends on it's value: X tends to zero at $\mu<1$, but approaches asymptotically a non-zero stable value at $1<\mu<3$ (higher parameter's values result in periodic and chaotic oscillations, but these values were never achieved in our simulation). At $\mu=1.151$ and in the absence of a noise, the stable value is equal to 0.13119... .

Fig. 5 shows how X changes when the noise is added to $\mu=1.151$. Values of X are now distributed around the stable value, but the latter is still most probable one provided σ is small. As the noise increases (σ increases), the distribution becomes asymmetric and small values are now most probable. This change of the distribution under the influence of noise can be described as the noise-induced phase transition (Horsthemke & Lefever, 1984).

The turn tendency tends to zero when the parameter decreases below 1 under influence of the noise; even if the parameter increases at the next step, the tendency

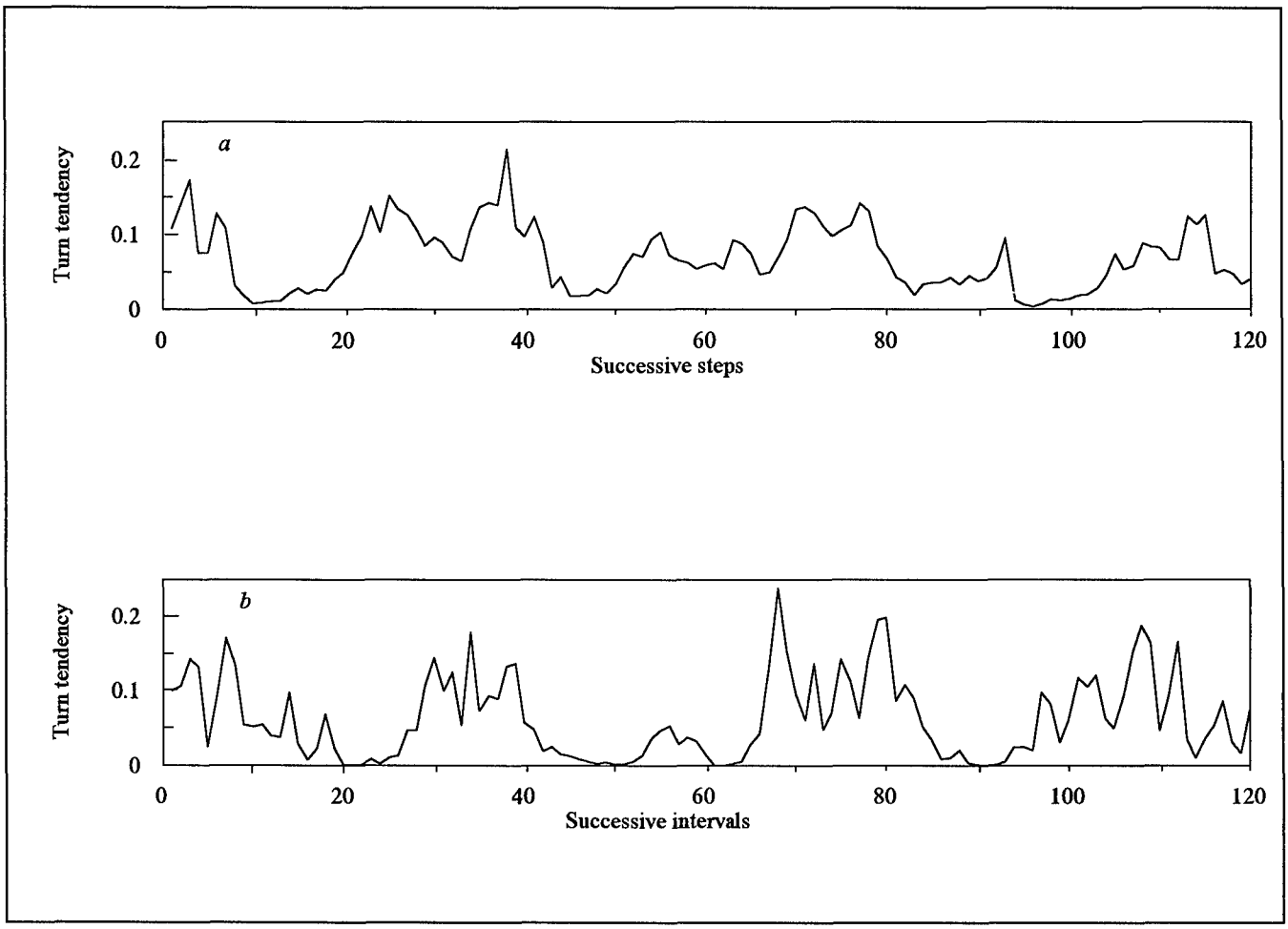


Fig. 4. (a) Turn tendency in the model at successive steps. (b) Average tendency in successive 10 steps intervals.

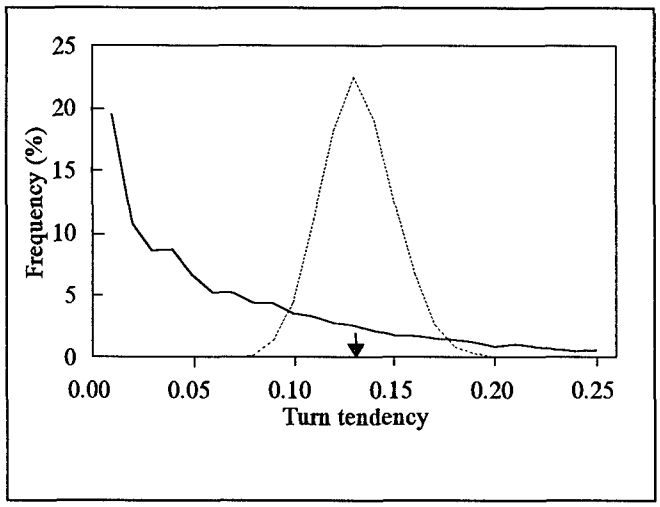


Fig. 5. Frequency distribution of turn tendencies in the model at $\sigma = 0.5$ (dotted line) and $\sigma = 2.3$ (line). Data are obtained from a single run (2000 steps). Value of turn tendency at $\sigma = 0$ is marked with arrow.

needs a time to increase again. On the other hand, once increased, the tendency would not decrease rapidly until the parameter falls abruptly. This behavior is accounted for by the influence of previous steps on subsequent ones, which results in the correlation between successive values of the tendency. Besides this influence, the alteration of series of high and low values depends on random variations of the parameter and so is not periodic.

5. Discussion

One may identify the carp behavior during series of turns as the restricted search, while runs between series may be viewed upon as the ranging. The problem is what makes carp switch between ranging and restricted search in our experiment. The alteration of two modes of behavior is unlikely to be a response to external landmarks inside or outside the corridor: any landmark should be repeatedly encountered by a fish swimming around the corridor while ranging, but evidently failed to evoke a response upon every encounter. Besides, no segment in the corridor was preferred by fishes for turning.

The other possibility is an internal or external pacemaker switching the restricted search on and off. The short-term periodicity in the swimming velocity and some other parameters of swimming behavior had been observed earlier in goldfish (Matis et al., 1973; 1974). We, however, failed to find any periodicity in turn frequency.

In our model, the observed behavior is the result of the phase transition induced by noise. While experimental records are too short for correct use of most methods of time series analysis and quantitative fitting of model's parameters to the carp behavior, our goal was to find the basic mechanism which may account for essential qualitative features of the data obtained. There is essential similarity between carp's and model's behavior. In both cases 1) there is correlation between successive behavioral variables and distinct alternation of two modes of behavior; 2) the correlation is scale-invariant to some extent, but decreases gradually as the scale increases; 3) the alteration of two modes is by no means regular. Thus, the model reproduces both orderliness in an uniform environment and irregularity we observed in the experiment.

Noise-induced transitions are widespread phenomena and had been observed in various systems. While we supposed the noise to be white Gaussian, it is by no means crucial for a transition. For instance, the modeling had shown that Markovian non-Gaussian voltage noise may cause phase transitions in the action potential generation when applied to an axon membrane (Horsthemke & Lefever, 1981). A possibility of a transition depends on many factors, but crucial ones are non-linearity of a system and presence of a noise (Horsthemke & Lefever, 1984). As both factors are characteristic to a nervous system, the possible role of the noise-induced transitions in the shaping of an animal behavior deserves to be investigated further. Above features (1-3) are not specific to fishes' exploratory behavior. Various activities in different animal species show scale-invariant switching between behavioral modes (Cole, 1995). Thus, our model may possibly have broader implication as a starting-point for an understanding of these activities. In fact, the conception of noise-driven behavior had been used earlier in human psychology to explain stochastic switching between alternative perceptual interpretations of ambiguous figures (Riani & Simonotto, 1994).

Noise and noise-induced transitions may be unavoidable in the nervous system and change the behavior notwithstanding their adaptive value. If so, it makes sense to incorporate these phenomena into the mechanism which controls behavior instead of suffering from them. When an animal explores an environment or searches for food or other objects it encounters several problems, and we will discuss now how the incorporation may help to solve these problems.

Firstly, an animal should detect a relevant external stimuli to initiate the restricted search. It was shown that a noise may help to amplify and process signals in sensory systems (Douglass et al., 1993; Braun et al., 1994; Wiesenfeld & Moss, 1995) and at cognitive level (Riani & Simonotto, 1994). In our model, an external stimulation can be simulated, for instance, by adding some value to the parameter λ. When this addition coincides in time with a high parameter value caused by the noise, one may expect the turn tendency to be amplified and, consequently, the probability of initiating the restricted search to rise. This may account for the fact that restricted search is more readily accomplished upon discovery of a food or food-related stimuli in the natural environment.

Secondly, consider an animal which starts to search for food upon discovery some food-related stimuli. Food is not necessarily to be found immediately. It pays to be persistent and search for a while even after the stimulation ceased. In the model, the phase transition and the correlation between turn tendencies are closely connected phenomena, and the correlation results in the persistence of behavior. If a stimulation is added to the parameter at nth step and results in high turn tendency, then the tendency and restricted search may persist at next steps also.

Thirdly, if there is no gain from the search, an animal should stop searching sooner or later. According to models of optimal foraging, the duration of the search should depend on the energy spent for it and expected profitability of the area searched (Stephens & Krebs, 1986). Animals, however, are not likely to make energy and profitability calculations, and rather simple 'rules of thumb' must exist to stop the search (Cheverton et al., 1985). The model provides such a rule: the search should cease when the turn tendency decreases under influence of the noise.

Lastly, the alteration of searches and runs resulted from the noise-induced transition may increase long-term efficiency of the searching behavior. Restricted searches in Fig. 1b are divided by relatively long runs with no turns. One may speculate that in more natural two-dimensional environment an animal will runs toward different directions between searches and these long runs prevent it from returning to areas already searched through and exhausted. Such a strategy should be more advantageous in comparison with the Brownian random walk.

While these considerations needs further experimenting and modeling to be verified quantitatively, they still show that the noise generated by nervous activity and noise-induced transitions may help to shape an adaptive behavior and make it more efficient instead of being a nuisance.

Acknowledgments

We would like to thank anonymous referee for helpful comments on the manuscript of this paper.

References

Aronson L.R. 1970. Functional evolution of the forebrain in lower vertebrates. In: Developmentand evolution of behaviour, pp.75-107. San Francisco: Academic Press.

Atema J., Todd J.H. & Bardach J.E. 1969. Olfaction and behavioral sophistications in fish. In: Olfaction and Taste, pp.241-251. New York: Academic Press.

Benhamou S. & Bovet P. 1989. How animals use their environment: a new look at kinesis. Anim.Behav., 38, 375-383.

Braun H.A., Wissing H., Schafer K. & Hirsch M.Ch. 1994. Oscillation and noise determine signal transduction in shark multimodal sensory cells. Nature, 367, 270-273.

Charnov E.L. 1976. Optimal foraging and the marginal value theorem. Theor. Pop. Biol., 9, 129-136.

Cheverton J., Kacelnik A. & Krebs J.R. 1985. Optimal foraging: constraints and currencies. In: Experimental Behavioral Ecology (ed. by B.Holldobler & M.Lindauer), pp.109-126. Stuttgart: G.Fisher Verlag.

Cole B.J. 1995. Fractal time in animal behaviour: the movement activity of Drosophila. Anim. Behav., 50, 1317-1324.

Coughlin D.J., Strickler J.R. & Sanderson B. 1992. Swimming and search behaviour in clownfish, Amphiprion perideraion, larvae. Anim. Behav., 44, 427-440.

Douglass J.K., Wilkens L., Pantazelou E. & Moss F. 1993. Noise enhancement of information transfer in craifish mechanoreceptors by stochastic resonance. Nature, 365, 337-340.

Fraenkel G.S. & Gunn D.L. 1961. The orientation of animals. Kineses, taxes and compass reactions. New York: Dover.

Gdovskii P.A., Gremyatchikh V.A. & Nepomnyashchikh V.A. 1994. Influence of anosmia on the glucose content and investigatory behaviour of the common carp in the presence of a visual stimulus. Zhurnal evol. i biochim. fiziologii., 30, 746-752 (in Russian).

Horsthemke W. & Lefever R. 1981. Voltage-noise induced transitions in electrically exitable membranes. Biophys. J., 35, 415.

Horsthemke W. & Lefever R. 1984. Noise-induced phase transitions. Berlin: Springer-Verlag.

Hunter J.R. & Thomas G.L. 1974. Effect of prey distribution and density on the searching and feeding behaviour of larval anchovy Engraulis mordax. In: The early life history of fish, pp.559-574. Berlin: Springer-Verlag.

Janzen W. 1933. Untersuchungen uber Grosshirnfunktionen des Goldfisches (Carassius auratus). Zool. Sahrb. Abt. allg. Zool. Physiol., 52, 591-628.

Kleerekoper H., Matis J., Gensler P. & Maynard P. 1974. Exploratory behaviour of goldfish Carassius auratus. Anim. Behav., 22, 124-132.

May R.M. 1976. Simple mathematical models with very complicated dynamics. Nature, 261, 459- 467.

Matis J., Kleerekoper H. & Gensler P. 1973. A time series analysis of some aspects of locomotor behavior of goldfish, Carassius auratus. J. interdiscipl. Cycle Res., 4, 145-158.

Matis J.H., Childers D.R. & Kleerekoper H. 1974. A stochastic locomotor control model for the goldfish (Carassius auratus). Acta Biotheoretica, 23, 15-54.

Munk P. & Kiorboe T. 1985. Feeding behavior and swimming activity of larval herring (Clupea harengus) in relation to density of copepod nauplii. Mar. Ecol. Prog. Ser., 24, 15-21.

Nepomnyashchikh V.A. & Gremyatchikh V.A. 1993. The relation between structure of trajectory and handedness of direction of locomotion in the Oreochromis mossambicus Peters (Cichlidae). Zhurnal obshchei biologii, 54, 122-129 (in Russian).

Nepomnyashchikh V.A., Gremyatchikh V.A. & Podgorny K.A. 1995. Orderliness and optimisation in animal behaviour. Uspekhhi sovremennoi biologii, 115, 432-438 (in Russian).

Riani M. & Simonotto E. 1994. Stochastic resonance in the perceptual interpretation of ambiguous figures: a neural network model. Phys. Rev. Lett., 72, 3120- 3123.

Scott W.B. & Crossman E.J. 1973. Freshwater fishes of Canada. Fish. Res. Board Canada Bull. 184, 1-966.

Segaar J. 1961. Telencephalon and behaviour in Gasterosteus aculeatus. Behaviour, 18, 256-287.

Stephens D.W. & Krebs J.R. 1986. Foraging Theory. Princeton, New Jersey: Princeton University Press.

Warburton K. 1990. The use of local landmarks by foraging goldfish. Anim.Behav., 40, 500-505.

Wiesenfeld K. & Moss F. 1995. Stochastic resonance and the benefits of noise: from ice ages to crayfish and SQUIDs. Nature, 373, 33-36.

Handling Time-Warped Sequences with Neural Networks

Claudia Ulbricht

Austrian Research Institute for Artificial Intelligence
Schottengasse 3, A–1010 Vienna, Austria
claudia@ai.univie.ac.at

Abstract

Being able to deal with time-warped sequences is crucial for a large number of tasks autonomous agents can be faced with in real-world environments, where robustness concerning natural temporal variability is required, and similar sequences of events should automatically be treated in a similar way. Such tasks can easily be dealt with by natural animals, but equipping an animat with this capability is rather difficult. The presented experiments show how this problem can be solved with a neural network by ensuring slow state changes. An animat equipped with such a network not only adapts to the environment by learning from a number of examples, but also generalizes to yet unseen time-warped sequences.

1 Introduction

For numerous tasks, autonomous agents have to be able to deal with time-warped sequences of events. Sequential patterns of variable length are common in real-world environments, and the number of available training examples are usually relatively small. Animats should not only be able to adapt to the given environment, but also automatically treat similar sequences of events in a similar way. They should be robust concerning natural temporal variability. Therefore, models capable of generalizing to time-warped patterns are needed. The neural network presented in this paper is shown to fulfil this requirement. Since it uses the input patterns for state formation, it is called the input state network. This model of temporal processing solves the given task by employing slowly changing states.

The task is described in Section 2. A brief overview of sequence processing with neural networks is given in Section 3. The two network models used in the experiments are presented in Section 4. In Section 5, the setup of the experiments is described, and the results can be found in Section 6.

2 The Task of the Agent

The Agent in its Environment: The task of the autonomous agent is to turn to the appropriate direction after having perceived a certain sequence of events. This can, for instance, be an animat which is situated in a maze and has to learn that the target (i.e. some reward like food) is to the left or right, depending on the characteristics of the aisle which it has passed. For instance, after having perceived a sequence like AAPD or AAPPD, the target is to the right, while it is to the left after sequences like XXPD or XXPPD. A and X represent arbitrary sources of sensory information, P denotes an aisle without any specific information, and D a decision point at an intersection. The input P models the pause between the input information, which is relevant to the decision, and the point in time when the decision is actually required.

Since passing some source (like A or X) takes some time, it is perceived several times in a row. The idea of repeating the sensory input is to simulate a real-world environment where a single appearance of an input is treated as noise. An input is not worth being noticed by the model unless it is present for some time and can be sampled several times in a row. The exact number of samples may vary, which may result in sequences like AAAPPPD or AAAAAPPD. These sequences are similar in that the overall order of sequence elements A,P, and D is maintained. Therefore, a new type of notation is introduced, where [X*] represents a sequence of an arbitrary number of sequence elements X. The sequences AAAPPPD and AAAAAPPD can thus be written as:

[A*] [P*] D.

The target of the autonomous agent is to the right after one of the sequences [A*][P*]D, [B*][P*]D, and [C*][P*]D, while it is to the left after one of the sequences [X*][P*]D, [Y*][P*]D, and [Z*][P*]D. The agent at such an intersection where a decision is required is depicted in Fig. 1.

The agent can perceive A, B, C, X, Y, Z, P, D, L, or R. It is trained to predict the location of the goal at

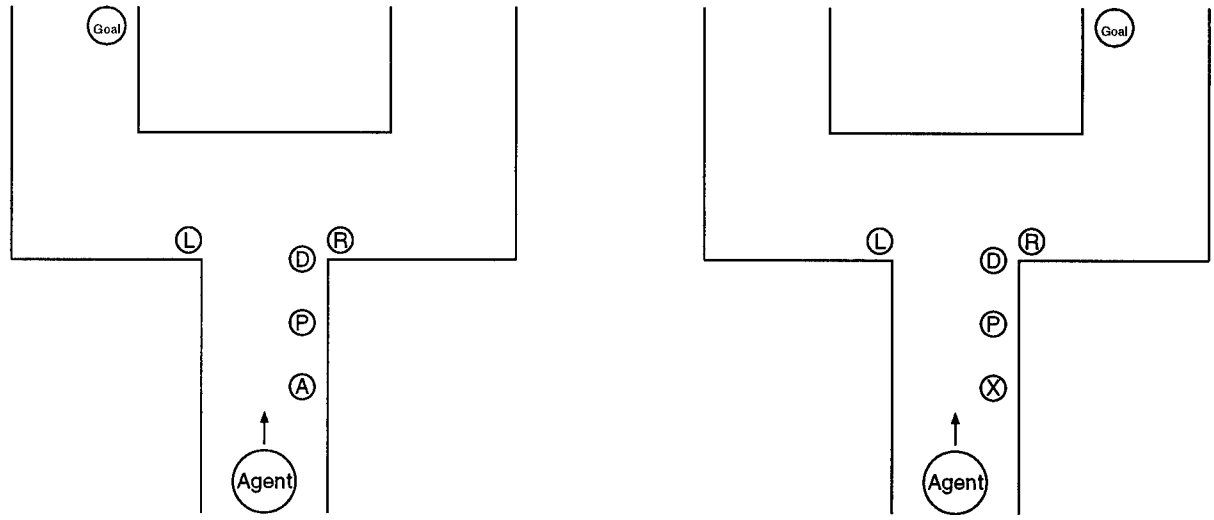


Figure 1: *The agent in its environment*

the correct point in time. The goal is hidden so that it cannot be perceived before passing the intersection. The outputs (L or R) denote turns to the left and right respectively. The pause P is used to tune the time span between information input and required output. It is needed for designing an environment where the distance between the relevant information and the required reaction is variable. The input pattern D denotes a decision point where the agent has to turn left or right. A typical input sequence consisting of three subsequences looks as follows:

AAAAPPPPPPPDLLLLLLLL
ZZZZZPPPPDRR
CCCCCCCCCPPDLLL .

In the concise format, this can be written as:

[A*][P*]D[L*][Z*][P*]D[R*][C*][P*]D[L*] .

This is the sequence of prototypical elements. The actual number of repeated samples varies dependent on the path followed by the agent and on the distance between the agent and the object. This variation is simulated by randomly determining how long each input is perceived.

The requirements of the task: The given environment requires a model which can memorize information for several time steps. The relevant criterion is the **length of the time period** between the information input and the point in time when this information is needed to produce the correct output. This time period can also be regarded as the **time lag** between an event and the desired response of the system. Long time lags, as they can be present in the given task, lead to **long input-output dependencies**.

The given task provides another challenge. It requires not only dealing with long dependencies, but also with **dependencies of variable length**. Since the sequence elements are samples taken at discrete intervals of time from a continuous environment, the sampling rate has no correspondence to any feature of the real-world environment. In such types of environment, subsequences which are slightly temporally displaced should be recognized as being similar and treated similarly unless this difference is actually relevant to the task, which it usually is not. In other words, **time-warped input patterns** should still be recognized. Designing models which are robust in respect to natural temporal variability is also the aim of the work presented in [Port and Anderson, 1989].

To summarize, the characteristics of realistic event sequences require a model capable of

- coping with dependencies of several time steps, and
- generalizing to time-warped patterns.

3 Neural Network Solutions

A sequence consists of sequence elements which are ordered in time. Usually, there is a single sequence element x per time step t:

$$\ldots, x(t-1), x(t), x(t+1), \ldots$$

Selecting an appropriate neural network is already difficult for static tasks, but when temporal tasks are to be handled, model selection becomes much more difficult. A task which involves only short dependencies can be handled more easily than one which involves long ones. Moreover, sequences should be treated similarly when they are stretched or squeezed with respect to the tem-

poral dimension. Handling such time-warped sequences is not trivial, because they are not automatically recognized as being similar.

Even though most neural network research has focused on processing single patterns, a large number of approaches to handling sequences have been developed and tested. Sequence processing requires a method for saving information for subsequent time steps. Detailed overviews of neural networks for handling temporal aspects are given, for instance, in [Ulbricht *et al.*, 1992], [Mozer, 1993], [Rohwer, 1994], and [Chappelier and Grumbach, 1994].

Here the following four methods are distinguished and briefly addressed:

1. Layer delay without feedback
2. Layer delay with feedback
3. Unit delay without feedback
4. Unit delay with feedback

These methods can be employed in different layers of the neural network. They can also be associated with different weights and delays of different length. Since they can be combined in various ways, the number of resulting combinations and thus actual network models, which can be used for handling sequences, are quite large.

The presented categorization scheme for sequence-handling methods is, in some parts, comparable to other taxonomies found in literature. The two categorization schemes provided by [Catfolis, 1994] and [Chappelier and Grumbach, 1994] refer mainly to the methods for sequence handling, whereas the scheme presented in [Mozer, 1993] covers also the trainability of network components.

1. Layer Delay without Feedback

The most straightforward approach is to use an **input window** which holds a restricted small number of past sequence elements. Then this part of the time series is analyzed before the window is shifted further in time by one or more elements. The same result can be achieved by delaying the contents of the input layer several times in a row. A network that contains no connections with time delays which feed information back so that loops emerge is called a **non-recurrent network**.

The complete mapping of a feedforward network from the input $\mathbf{i}(t)$ to the output $\mathbf{o}_1(t)$ can be written as:

$$\mathbf{o}_1(t) = \mathbf{F}_1(\mathbf{i}(t)), \tag{1}$$

where $\mathbf{F}_1$ is the mapping of a neural network with a hidden layer, which is also called a "multi-layer perceptron."

The output $\mathbf{o}_2(t)$ of a network with an input window is

$$\mathbf{o}_2(t) = \mathbf{F}_2\left(x(t-1), x(t-2), \ldots, x(t-p)\right), \tag{2}$$

where p is the window size, and $x(t-1), x(t-2), \ldots, x(t-p)$ are the elements of the input sequence. This is comparable to auto-regressive models (AR-models) of order p, as they are treated in [Box and Jenkins, 1970]. Apart from the non-linear mapping, the two models are identical.

2. Layer Delay with Feedback

Layers can be delayed and fed back to previous layers in the updating order. The resulting **cycles** in such **recurrent networks** allow the output of a unit to return to the same unit at a later point in time. Hidden and output layer feedback are the most common forms of layer delay with feedback.

If recurrent networks are updated like feedforward networks with a single update cycle per time step, they keep the general characteristics of feedforward networks. Models of this kind are also called "simple recurrent networks," "recurrent feedforward networks," or "partially recurrent networks" [Hertz *et al.*, 1991].

The context provided by the feedback loop can also be regarded as the **state** the network is in. Therefore, the layer keeping the delayed information is also referred to as the "state layer." This idea of state is comparable to that in state automata. There, the new state is solely dependent on the previous state and the last input. The state subsumes the input history. This is related to the Markov property, which ensures that all the information which is relevant to the future is contained in the present state.

In a simple recurrent network with hidden or output layer feedback, the network output is a function of the input $\mathbf{i}(t)$ and the state $\mathbf{s}_1(t)$ of the network:

$$\mathbf{o}_3(t) = \mathbf{F}_3\left(\mathbf{i}(t), \mathbf{s}_1(t)\right). \tag{3}$$

Typically, the input consists of a single sequence element:

$$\mathbf{o}_3(t) = \mathbf{F}_3\left(x(t-1), \mathbf{s}_1(t)\right). \tag{4}$$

The next state $\mathbf{s}_1(t)$ is dependent on the input $\mathbf{i}(t-1)$ and on the most recent state $\mathbf{s}_1(t-1)$:

$$\mathbf{s}_1(t) = \mathbf{g}_1\left(\mathbf{i}(t-1), \mathbf{s}_1(t-1)\right). \tag{5}$$

Here $\mathbf{g}_1$ stands for some unknown function. When replacing $\mathbf{s}_1(t)$ in Equation 3 several times, it can be seen that the output $\mathbf{o}_3(t)$ is dependent on several $(n+1)$ inputs $\mathbf{i}(t), \mathbf{i}(t-1), \ldots, \mathbf{i}(t-n)$:

$$\mathbf{o}_3(t) = \mathbf{g}_2\left(\mathbf{i}(t), \mathbf{i}(t-1), \ldots, \mathbf{i}(t-n), \mathbf{s}_1(t-n)\right), \tag{6}$$

where $\mathbf{g}_2$ is another function which is not further specified. Equations 3 and 5 represent the two mappings which are part of this type of network:

(a) the feedforward mapping of all the available information ($\mathbf{i}(t)$ and $\mathbf{s}_1(t)$) to the output $\mathbf{o}_3(t)$, and

(b) the next-state function modeling the mapping from the past state $\mathbf{s}_1(t-1)$ and additional information to the current state $\mathbf{s}_1(t)$.

In addition to the input, two different entities influencing the output can be distinguished:

(a) the weights of the feedforward connections, and

(b) the activations of the state units.

These play different roles. Typically, the unit activations of the state layer change with each time step, while the weights are adjusted at a much lower pace. The weights represent the long-term memory holding all the acquired knowledge. The state is needed for processing the preceding sequence elements. It provides the context for the new input.

3. Unit Delay without Feedback

The units themselves can also have temporal properties. A unit without feedback can be created by delaying information within the unit for a limited number of time steps. However, this approach is not very common.

4. Unit Delay with Feedback

If the activation of a unit is influenced by its own preceding activation, this can be modeled by a feedback loop, which can be referred to as "**self-recurrent feedback loop**." In contrast to layer feedback, one can speak of "**self-recurrent unit feedback**." The self-recurrent feedback loops carry weights like the other connections in the network. A unit with a feedback loop is depicted in Fig. 2. It receives its own preceding output in addition to the input coming from other units.

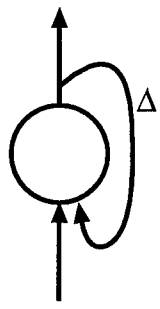


Figure 2: *Unit delay with feedback*

The activation of a unit $z_k(t)$ with feedback (the k-th unit in layer z) may be determined by adding its own preceding activation to the value coming in from another layer (for instance, from the input layer):

$$z_k(t) = \sigma\left(\sum_{j=1}^{J} i_j(t)\, w_{jk}\right) + z_k(t-1). \qquad (7)$$

where the w_{jk} are the weights linking units j and k.

Model Selection: All discussed networks can be used for processing sequences. They take into account the order of the sequence elements, and—if designed appropriately—can deal with long input-output dependencies. However, not all are well suited to coping with dependencies of variable length. They do not automatically treat patterns which are similar but warped in time similarly. Sequence elements arriving earlier or later than usual distract these systems. Such distractions might be due to missing or superfluous sequence elements, or simply due to changes in speed. This problem is called the **temporal invariance problem**.

Since non-recurrent networks parallelize temporal information, they do not automatically generalize to time-warped patterns. Recurrent networks are more flexible concerning the temporal dimension. However, most networks—including simple recurrent networks with layer feedback—are rigid in the sense that the lags between the sequence elements have to be of exactly the same length to be treated similarly. One of the reasons is that the state changes completely with each time step. Since the values returned by the feedback loops differ from the original values, consecutive states are not similar. In other words, states close in time have little in common and thus they cannot be treated similarly.

One solution to the temporal invariance problem is to represent time explicitly. For instance, in [Mannes, 1992], a model based on the ART network processes sequences by treating the order of sequence elements and timing information separately. However, if time should be handled implicitly, another solution has to be found.

Slowly Changing States: The solution to the temporal invariance problem proposed here is to use network states which move slowly through state space. The problem with models which do not produce such **sluggish states** is that the network's walk through state space is not automatically smooth. Subsequent states may lie far apart. This is due to the mapping from one state to the next which does not assure that subsequent states are similar. Thus, subsequent states are related, but not similar to each other. This causes the problem that input, which occurs a single time step earlier or later than usual, leads to a completely different output. This is usually not wanted when the input is sampled at time

lags which are independent of the actual sequence. This problem can be solved by forcing the next-state function, which models the relationship between consecutive states, to perform a nearly identical mapping. This aspect is also mentioned in [Jordan, 1986]:

> *State vectors at nearby points in time must be similar.*

The resulting "**continuity property**" is necessary in order to obtain slowly varying states. This requirement is usually not taken into account. The **requirement of state continuity** ensures a slowly changing state. On the other hand, the state must not remain stable, because two consecutive states have to differ with respect to the most recent sequence element. Therefore, consecutive states in a network are forced to be similar, but distinguishable. This way, warped inputs can be handled similarly *or* differently, according to what is required for the given task. Such sluggish states can be obtained by using a **nearly auto-associative next-state function**. This can be achieved, for instance, by inserting **self-recurrent feedback links** from state units to themselves.

Memories modeled with self-recurrent feedback loops represent just one potential approach to obtaining similar subsequent states. Other nearly auto-associative next-state functions may also be applicable. The important point is that the next-state function has to be auto-associative.

The strength of employing sluggish states is that such a model automatically **generalizes to time-warped patterns**. It is also well suited to handling **noisy input**, because its state is quite robust. In other words, the state is not quickly distracted by single outliers. A problem, though, which cannot be solved by such sluggish units, is that recent events have more influence on the state than events in the distant past. The memory decays at a slower rate than when there are no self-recurrent feedback loops, but it still decays. Arbitrary time lags could be bridged if there were a way to obtain temporary variable bindings. This area might be another chance for neural network research to learn from biological models.

Self-recurrent Feedback: Due to additional self-recurrent feedback loops, the state vector activations change more slowly. Thus, time-warped and noisy input is handled automatically. Such a network, with a state wandering slowly through the state space, has the intrinsic capability of handling temporal dependencies in spite of time-shifted and otherwise disrupted input. In the taxonomy presented in [Mozer, 1993], such memory layers with self-recurrent feedback loops are referred to as "exponential trace memories" because they decay exponentially with time. The resulting memory contents represent an exponentially weighted average of the input sequence. The speed of decay is dependent on the weights of the self-recurrent feedback loops.

In contrast to pure copies of layers, such memories with self-recurrent feedback loops are of low resolution. Global aspects can be detected more easily and dependencies spanning long time periods can be recognized. The resulting representation is a **reduced description** [Mozer, 1992] of the sequence. This type of smoothing is also comparable to low-pass filtering and sampling at a lower rate.

The requirement of state continuity is met, for instance, in the network described in [Jordan, 1986]. There, an exponentially weighted average of past network *outputs* is produced. This way, all arbitrarily distant past outputs have some influence on the state. However, the strength of this influence decreases quickly with time. The continuity of the state is achieved by feeding back the activation of each state unit to itself. The previous output unit activations are simply added to these state activations with weights of 1:

$$\mathbf{s}_2(t) = \mu_1\, \mathbf{s}_2(t-1) + \mathbf{o}_4(t-1), \tag{8}$$

where $\mathbf{o}_4(t-1)$ is the result of mapping the input to the output, the output of the network at time $t-1$, and μ_1 is a real-valued constant ($0 < \mu_1 < 1$) modeling the weight, which modifies the feedback of the state units to themselves. This value can be tuned to make the state more or less flexible. As a result, adjacent states are more or less similar. When setting the first state equal to the initial network output ($\mathbf{s}_2(1) = \mathbf{o}_4(0)$), the state can also be described by the following formula:

$$\mathbf{s}_2(t) = \sum_{r=1}^{t} \mu_1^{r-1} \mathbf{o}_4(t-r). \tag{9}$$

This shows that the current state is a function of all past network outputs. The similarity of adjacent states is the result of averaging past states. It is difficult to determine the appropriate value for μ_1, though. Therefore, [Jordan, 1986] proposes to let the system learn the appropriate "next-state function."

Input State: The novel neural network presented in Section 4 employs self-recurrent feedback for creating the state of the network. This way, the state is directly influenced by the original inputs. The network described in [Jordan, 1986] uses the network outputs to produce the state, but here the inputs are forwarded to the state layer. Another difference is that the incoming values (in this case $\mathbf{i}(t-1)$) are weighted by $1-\mu_2$, as shown in the following equation:

$$\mathbf{s}_3(t) = \mu_2\, \mathbf{s}_3(t-1) + (1-\mu_2)\, \mathbf{i}(t-1). \tag{10}$$

This ensures that the sum never exceeds 1 and permits the weights to be interpreted as the degree of influence. They can thus be turned into statements in percent.

4 Tested Neural Network Models

As a result of the discussion in the preceding section, two different network models were selected for being tested in this study:

1. a simple recurrent network with hidden layer feedback, and
2. the input state network.

1. The Simple Recurrent Network with Hidden Layer Feedback

A simple recurrent network with hidden layer feedback, like the one presented in [Elman, 1990], is depicted in Fig. 3. While the dashed arrows denote 1:1-copies of layers, the full arrows represent m:n-connections, where all m units are linked with all n units in the other layer.

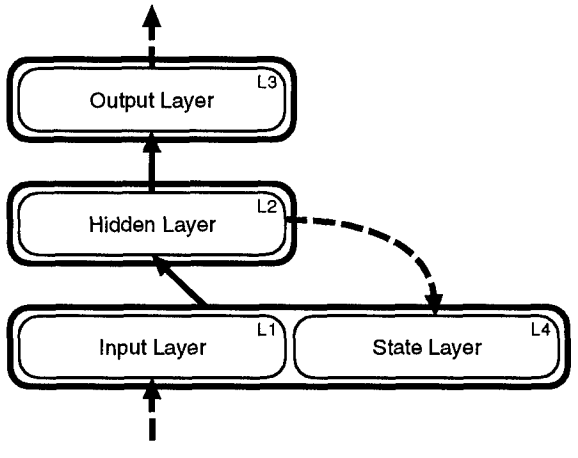


Figure 3: *Recurrent network with hidden layer feedback*

The output of such a network is dependent on the input and the preceding hidden layer activations:

$$\mathbf{o}_5(t) = \mathbf{F}_4\left(\mathbf{i}(t), \mathbf{h}_1(t-1)\right). \qquad (11)$$

This equation does not show an important property of $\mathbf{h}_1(t-1)$, namely that it typically represents past inputs in a format which is relevant with respect to the task of the model. Therefore, hidden layer feedback often leads to good results.

The activation of a unit s_{4_j} is

$$s_{4_j} = (1-\mu_3)h_{1_j}(t) + \mu_3 s_{4_j}(t-1), \qquad (12)$$

where h_{1_j} is a unit j in the hidden layer $h_1(t)$. Written in the recurrent form, the activation of a unit k in this layer is:

$$h_{1_k}(t) = \sigma\Big(\sum_{j=1}^{J} i_j(t)\, v_{jk} + \sum_{l=1}^{L} h_{1_l}(t-1)\, u_{lk}\Big), \qquad (13)$$

where the weight v_{jk} leads from the unit j in the input layer to the unit k in the hidden layer, and the weight u_{lk} is the weight of the self-recurrent feedback loop linking unit l and unit k in the hidden layer.

2. The Input State Network

The input state network resembles a typical multi-layer perceptron trained by backpropagation. It also has two layers of weights connecting three layers of units. However, the original input is only used to form the state. Due to the self-recurrent feedback loops, subsequent states tend to be similar. The contents of the state layer deal as input to the conventional multi-layer perceptron. Since the preceding state and the new input are combined to form the state of the network, the first layer of the multi-layer perceptron is called the "state layer," and the network is called the "input state network." The network is depicted in Fig. 4. It is close to the network models described in [Mozer and Soukup, 1991] and [Mozer, 1994], but it differs in several respects. One important difference is that, in these networks, the weights of the connections leading to the state layer are trainable. It is also different from the network presented in [Mozer, 1992], where the units in the *hidden* layer are equipped with temporal properties.

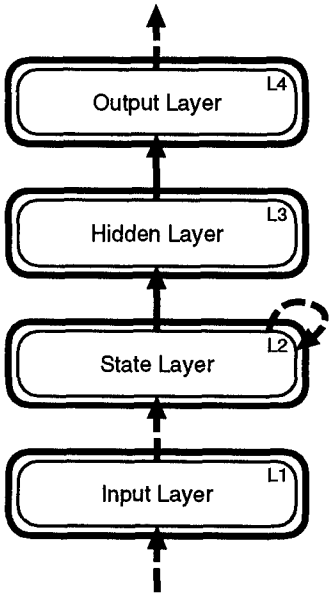


Figure 4: *The input state network*

Like in a network with hidden layer feedback, the new state $\mathbf{s}_5(t)$ is a function of the network input $\mathbf{i}(t)$ and the previous state $\mathbf{s}_5(t-1)$:

$$\mathbf{s}_5(t) = \mathbf{g}_3\left(\mathbf{i}(t), \mathbf{s}_5(t-1)\right), \qquad (14)$$

where $\mathbf{g}_3$ is some function which is not further specified at this point. The output $\mathbf{o}_6(t)$ is simply a func-

tion of the state:

$$\mathbf{o}_6(t) = \mathbf{F}_5\left(\mathbf{s}_5(t)\right), \tag{15}$$

where $\mathbf{F}_5$ is the mapping produced by a multi-layer perceptron. Recent inputs are implicitly included, because they contribute to $\mathbf{s}_5(t)$:

$$\mathbf{o}_6(t) = \mathbf{F}_5\left(\mathbf{g}_3\left(\mathbf{i}(t), \mathbf{s}_5(t-1)\right)\right). \tag{16}$$

The activation of a unit in the state layer $s_{5_j}(t)$ is determined as follows:

$$s_{5_j}(t) = (1-\mu_{4_j})\; i_j(t) + \mu_{4_j}\; s_{5_j}(t-1), \tag{17}$$

where the $i_j(t)$ are the unit activations in the input layer, and the μ_{4_j} are the weights of the self-recurrent feedback loops of the state units $s_{5_j}(t)$.

State Formation: In the input state network, each single unit in the state layer is given a different **flexibility**. The weights of the self-recurrent feedback loops μ_{4_j} range from zero to close to one, with fewer low and more high weights. Such a distribution is obtained by using the following formula for determining the weight μ_{4_j} of the feedback loop of the j-th unit:

$$\mu_{4_j} = \begin{cases} 0 & \text{if } j = 1 \\ 1 - \sqrt{\frac{j-1}{J}} & \text{otherwise,} \end{cases} \tag{18}$$

where J is the total number of units in the state layer, and j is the number of the unit in the state layer starting with 1. The connections from the input to the state layer are weighted by $(1-\mu_{4_j})$, so that the resulting activations range again from 0 to 1. This is displayed in Fig. 5 with $J = 20$. The result is a collection of state units of different flexibility, which form a set of short-term memories of different length. This collection of different memories is comparable to that in multi-recurrent networks, which are described in [Ulbricht, 1994]. Due to the flexible unit without any self-recurrent feedback loop (i.e. the first one with a feedback loop of zero weight), current input is quickly recognized. This is important, for instance, for input D, which demands a quick reaction.

Preprocessing: Since each unit $s_j(t)$ in the state layer has different temporal characteristics due to the different weights μ_{4_j} of the self-recurrent feedback loops, it is not reasonable to encode the input using single active units as, for instance, the vector (1000000000) for input A. This way, different sequence elements would be weighted by separate weights. In order to subject different inputs to various types of state units with different weights μ_{4_j},

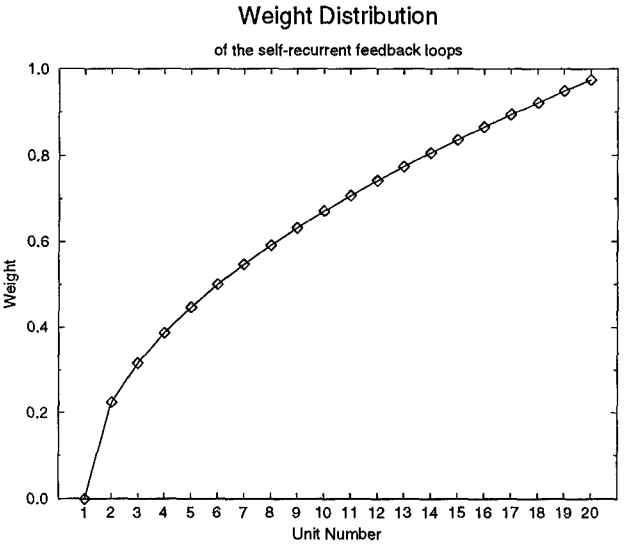


Figure 5: *Weight distribution in the state layer*

a distributed input representation is used. It is obtained by mapping the vectors with N single active units $a_n(t)$ via a set of weights b_{nm} and a sigmoid function to the M units $c_m(t)$:

$$c_m(t) = \sigma \sum_{n=1}^{N} a_n(t)\; b_{nm}, \tag{19}$$

where the weights b_{nm} are taken from a randomly selected set of weights. The layers are not fully connected, but only 80% of the connections are actually used. The sigmoid function

$$\sigma(x) = \frac{1}{1 + e^{(-d \cdot x)}}, \tag{20}$$

with a d of 20 is used to obtain the input distribution. Due to the sigmoid function, the activations $c_m(t)$ lie again in the range [0,1].

The employed technique of preprocessing the input provides a method for obtaining a distributed representation. There is evidence that in natural systems, information is also represented in a distributed format. As can, for instance, be shown in experiments with rats, the hippocampus seems to use an "ensemble code" for location information [Wilson and McNaughton, 1994].

Properties: In the input state network, the self-recurrent feedback loops are used to fulfil the continuity requirement described in Section 3. The strength of this network model is that the actual input is delayed and combined with the new input at the next time step. Thus the original input information can be memorized. For the task described here, this solution is appropriate, because there is more information

in the original input than in the hidden or output layer activations. This is due to the fact that preceding outputs are not relevant. For numerous other applications, though, self-recurrent feedback alone is too weak. In some cases, additional hidden and output layer feedback may be appropriate, or other techniques like layer delay or layer feedback may be required. Which type of delay is appropriate is dependent on the given task, but keeping some of the original input information is very likely to be useful.

5 Experiment Setup

The two networks presented in the preceding section are used for the experiments. The input state network has 20 input units, 5 hidden units, and 2 output units, plus bias units in the input layer and the hidden layer. The simple recurrent network has the same number of units, plus 5 state units for the past hidden unit activations.

Training: The networks are trained to perform the autonomous agent task described in Section 2. While the mapping functions (Equations 11 and 15) are trained by error-backpropagation, the next-state functions (Equations 12 and 14) are fixed. In order to evaluate the results, the mean square error (MSE) is determined:

$$\mathrm{MSE} = \frac{1}{N} \sum_{n=1}^{N} (y_a(n) - y_d(n))^2 , \qquad (21)$$

where $y_a(n)$ is the n-th network output, and $y_d(n)$ the n-th target output.

The subsequences used for training the agent to select the correct action at the intersection are selected in random order. For testing, a fixed set of 44 subsequences is used. During training, the agent slowly adapts to its environment. The weights of the randomly initialized neural network are adjusted based on the presented sequences of sensory input. This adjustment belongs to the type which is denoted "adaptation by learning" by [McFarland, 1991].

Testing the Generalization Capability: Since the generalization over the temporal dimension is to be investigated, the length of the pause between the relevant input and the required decision is set to a fixed value during training by putting a fixed number of inputs P in a row. For the experiments, the pause length was set to three time steps. In the end, the network generalizes over the temporal dimension *even though* the time span has not been varied during training.

In order to solve this task, the networks have to use information which was obtained several time steps back in time. Such a task can easily be handled with an input window or with time delays of appropriate length, if the length of the time lags does not vary. However, they cannot generalize over the temporal dimension, as required for solving the given task. If the input arrives a little earlier or later—for instance, after 2 or 4 time steps—these networks would not automatically treat the input similarly.

6 Results

Network with Hidden Layer Feedback: The tested networks with hidden layer feedback did not learn to solve the task. The results obtained with one of them are displayed in Figures 6 and 7. The network produced a similar output at all decision points. The activation of one output unit was close to one, and that of the other close to zero. For all tests with varying pause length, the MSE is similar, because 50% of the decisions were correct, and 50% were wrong.

Several variants of networks with hidden layer feedback—including networks with self-recurrent feedback loops in the state layer which holds the preceding hidden layer activations—were trained on the given task, but the result was the same. However, this does not imply that the underlying network model is inappropriate, the failure may be due to the weakness of the training procedure.

Input State Network: The experiment with the input state network was repeated several times. The variance was rather small. A typical result is visualized in Fig. 8. The mean square error is low in the neighborhood of the pause length used during training. In this case, the results for a pause length of 4 and 5 even happen to be slightly lower than for a pause length of 3.

A higher mean square error is not automatically equal to an incorrect output. Therefore, the percentage of correct decisions in several test sets with varied pause length is shown in Fig. 9. For pauses shorter than 2 and longer than 6, the network makes some mistakes, but for a pause length from 2 to 6, the network still comes up with a correct response in all the test problems. These results indicate that the network successfully learned to handle the given task. It is indeed capable of generalizing to time-warped sequential patterns.

7 Conclusion

Neither window networks nor simple recurrent networks with layer feedback are applicable to the given problem which is based on handling time-warped sequences. This is not only due to the network models, but also to the employed training algorithms. Such tasks are common in real-world environments and coping with them is easy for natural animals, but many neural networks cannot deal with them. The experiments have shown how the input state network can be trained to solve such a task. In this model, the original inputs are used to

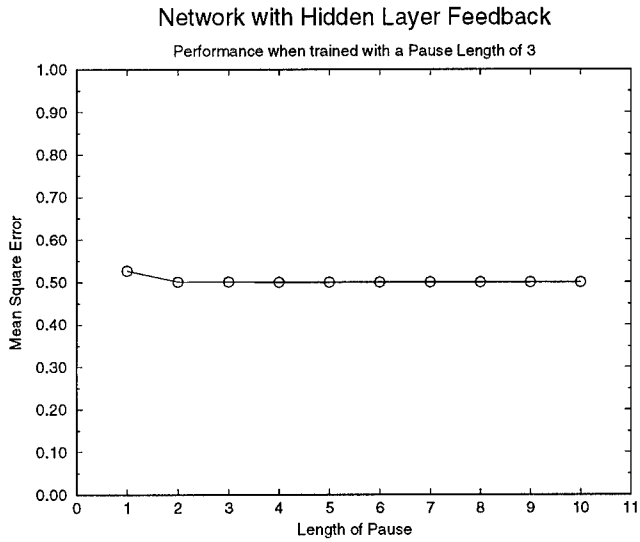


Figure 6: *Error of a network with hidden layer feedback for test sets with different pause length*

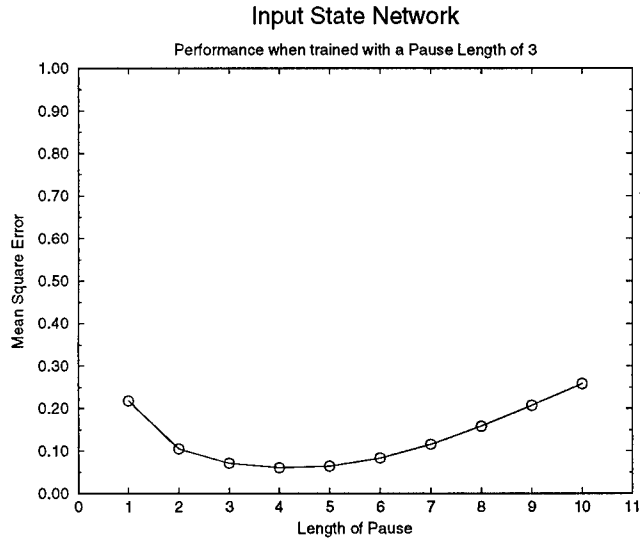


Figure 8: *Error of the input state network for test sets with different pause length*

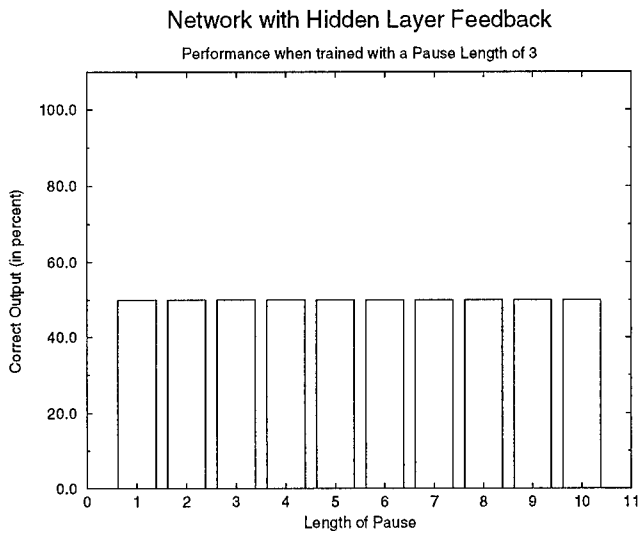


Figure 7: *Performance of a network with hidden layer feedback for test sets with different pause length*

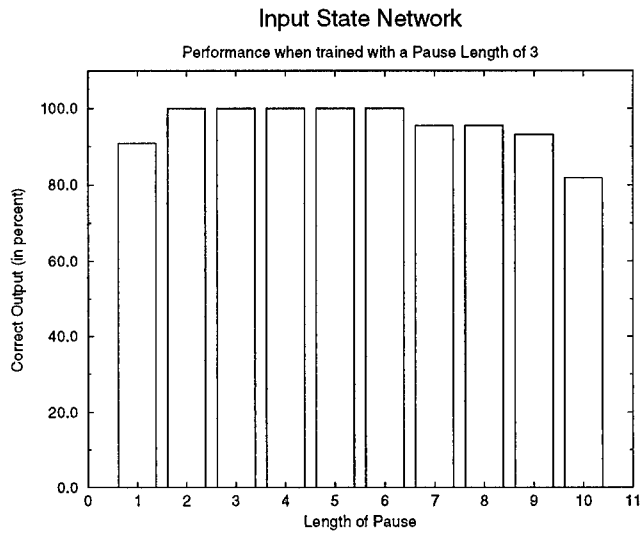


Figure 9: *Performance of the input state network for test sets with different pause length*

form the state. A single mechanism is employed: self-recurrent unit feedback. The resulting slowly changing states have been shown to enable the animat to deal with time-warped forms of event sequences. This method is well suited to modeling short-term memories of different length within a single system. Moreover, it supports generalization over the temporal dimension, which is crucial for autonomous agents faced with environments containing time-warped sequences of events. The robustness concerning natural temporal variability is an intrinsic property of this model of temporal processing. Even if the network is always trained with a sequence of fixed length, it treats time-warped forms of this sequence similarly. If time-warped versions are part of the training set, this task is even acquired much more easily. To conclude, models employing slowly changing states are useful for handling realistic sequences of sensory events. Thus, they are important for autonomous agents when their tasks involve the treatment of time-warped sequences.

Acknowledgements

Thanks for valuable comments are due to Erich Prem and Georg Dorffner.

The Austrian Research Institute for Artificial Intelligence is supported by the Austrian Federal Ministry of Science, Transport, and the Arts.

References

[Box and Jenkins, 1970] G.E. Box and G.M. Jenkins. *Time Series Analysis.* Holden-Day, San Francisco, 1970.

[Catfolis, 1994] T. Catfolis. Mapping a Complex Temporal Problem into a Combination of Static and Dynamic Neural Networks. *SIGART Bulletin, Vol. 5, No. 3*, 1994.

[Chappelier and Grumbach, 1994] J.-C. Chappelier and A. Grumbach. Time in Neural Networks. *SIGART Bulletin, Vol. 5, No. 3*, pages 3–11, 1994.

[Elman, 1990] J.L. Elman. Finding Structure in Time. *Cognitive Science*, 14:179–211, 1990.

[Hertz *et al.*, 1991] J. Hertz, A. Krogh, and R.G. Palmer. *Introduction to the Theory of Neural Computation.* Addison-Wesley Publishing Company, 1991.

[Jordan, 1986] M.I. Jordan. Attractor Dynamics and Parallelism in a Connectionist Sequential Machine. In *Proceedings of the Eighth Annual Conference of the Cognitive Science Society*, pages 531–546. Erlbaum, Hillsdale, NJ, 1986.

[Mannes, 1992] C. Mannes. A Neural Network Model of Spatio-Temporal Pattern Recognition, Recall and Timing. In *Proceedings of the IJCNN International Joint Conference on Neural Networks, Baltimore, IEEE*, pages 109–114, 1992.

[McFarland, 1991] D. McFarland. What It Means For Robot Behaviour To Be Adaptive. In J.-A. Meyer and S.W. Wilson, editors, *From Animals to Animats.* A Bradford Book, MIT Press, Cambridge, MA, 1991.

[Mozer and Soukup, 1991] M.C. Mozer and T. Soukup. Connectionist Music Composition Based on Melodic and Stylistic Constraints. In R.P. Lippmann et al., editors, *Advances in Neural Information Processing 3*, pages 789–796. Morgan Kaufmann, San Mateo, 1991.

[Mozer, 1992] M.C. Mozer. Induction of Multiscale Temporal Structure. In J.E. Moody, editor, *Neural Information Processing Systems 4, Morgan Kaufmann, San Mateo, CA*, 1992.

[Mozer, 1993] M.C. Mozer. Neural Net Architectures for Temporal Sequence Processing. In A. Weigend and N. Gershenfeld, editors, *Predicting the Future and Understanding the Past.* Addison-Wesley Publishing, Redwood City, CA, 1993.

[Mozer, 1994] M.C. Mozer. Neural Network Music Composition by Prediction: Exploring the Benefits of Psychoacoustic Constraints and Multiscale Processing. *Connection Science*, 1994.

[Port and Anderson, 1989] R. Port and S. Anderson. Recognition of Melody Fragments in Continuously Performed Music. In *Proceedings of the Eleventh Annual Conference of the Cognitive Science Society*, pages 820–827. Lawrence Erlbaum, Hillsdale, NJ, 1989.

[Rohwer, 1994] R. Rohwer. The Time Dimension of Neural Network Models. *SIGART Bulletin, Vol. 5, No. 3*, pages 36–44, 1994.

[Ulbricht *et al.*, 1992] C. Ulbricht, G. Dorffner, S. Canu, D. Guillemyn, G. Marijuán, J. Olarte, C. Rodriguez, and I. Martin. Mechanisms for Handling Sequences with Neural Networks. In C.H. Dagli et al., editors, *Intelligent Engineering Systems through Artificial Neural Networks, ANNIE'92*, volume 2, pages 273–278. ASME Press, New York, 1992.

[Ulbricht, 1994] C. Ulbricht. Multi-recurrent Networks for Traffic Forecasting. In *Proceedings of the AAAI'94 Conference, Seattle, Washington*, volume II, pages 883–888, 1994.

[Wilson and McNaughton, 1994] M.A. Wilson and B.L. McNaughton. Reactivation of Hippocampal Ensemble Memories During Sleep. *Science*, 265, 1994. 29. July 1994.

INTERNAL WORLD MODELS AND NAVIGATION

How insects learn about the sun's course: alternative modeling approaches

Jeffrey Dickinson

Fred Dyer*

Department of Zoology
Michigan State University
East Lansing, MI 48824
dickins9@pilot.msu.edu
fcdyer@msu.edu

Abstract

One of the major puzzles in animal behavior, and a major problem to be solved in the design of robots, concerns how spatial patterns in the environment can be encoded internally and used for navigation in a complex natural environment. Most work on this issue has concerned landmark learning. This paper deals with a phenomenon of spatial learning that is at least as widespread in the animal world as landmark learning, but has received comparatively little attention. The phenomenon is the ability to learn the course of the sun relative to earth-bound features, and thus to use the sun as a true compass. After reviewing behavioral evidence from bees and ants, two particularly well studied species, we evaluate the applicability of symbolic and connectionist approaches to modeling the internal representation of this environmental pattern.

1. Learning and the Sun Compass

The sun is an ideal reference for holding a straight course through the environment because it is at essentially infinite parallax distance. Thus, an animal can stay on course simply by maintaining a fixed body angle relative to the sun's azimuth. To use the sun over an extended period of time, however, requires compensating for solar movement relative to the preferred direction of travel. Complicating this task is that the azimuth changes at a variable rate over the day, shifting relatively slowly as the sun rises in the morning and sets in the evening, and relatively quickly when the sun is high in the sky around midday (Fig. 1A). Furthermore, the function describing the change of the azimuth as a function of time, or the solar ephemeris function, varies with season and latitude (Fig. 1B). Accurate sun-compass orientation depends upon the animal being informed of the current local ephemeris function (Gallistel 1990; Dyer and Dickinson in press).

We have known since the 1950s that many animals learn the current solar ephemeris function early in life, somehow integrating information regarding the sun's position relative to earth-bound features of the environment with information about time of day (from the animal's internal clock, which is entrained by the local light-dark cycle). One of the most intriguing findings of the early studies was that honey bees and other animals can roughly estimate the sun's position at times of day or night when they have never seen it. Thus, the learning mechanisms, whatever they are, do not merely compile the equivalent of an internal lookup table recording time-linked solar positions that the animal has had the opportunity to measure. Instead, they can also compute unknown positions of the sun. The puzzle is how such a computation could be carried out in the brain.

In a recent review of this problem, Gallistel (in press) suggested that the sun-learning abilities of insects could illuminate a fundamental question about how brains represent the world. Specifically, the question is whether, as traditionally assumed, a brain represents environmental patterns by the equivalent of mathematical computations operating on neurally encoded symbols of relevant stimuli. The alternative is the more recently developed idea that

*Author for correspondence

environmental patterns are represented in the patterns of synaptic weights distributed across a large number of parallel units in a neural network. Such a distributed representation may extract important structural features of patterns presented to the sense organs, but it does so without an explicit mathematical description of the represented pattern. The representation is said to be encoded "subsymbolically" rather than symbolically (Smolensky 1988).

Gallistel argued strongly that sun learning by insects and other animals cannot easily be explained in a connectionist framework, and that we must instead look for the neurobiological equivalent of mathematical operations on inputs symbolizing solar position and time of day. He asked, skeptically, "what does a non-symbolic ephemeris function look like?"

Gallistel's formulation of this problem raises two issues, which we address in this paper. First, if an insect's (or any other animal's) representation of the sun's course does indeed involve processes resembling computations on neurally encoded symbols of solar position and time of day, then what specific computations do we need to invoke to account for the animal's behavior? Thus, turning Gallistel's question around, what does a *symbolic* ephemeris function look like? Second, accepting Gallistel's challenge, is it possible to account for the behavior (as Gallistel argued it is not) within a subsymbolic connectionist framework?

An important step in evaluating these issues is to consider behavioral evidence regarding the performance of whatever mechanism underlies sun learning. Indeed, behavioral data are the only data we have at this point. As the next section should make clear, however, the behavioral evidence imposes strong constraints on models of sun-learning.

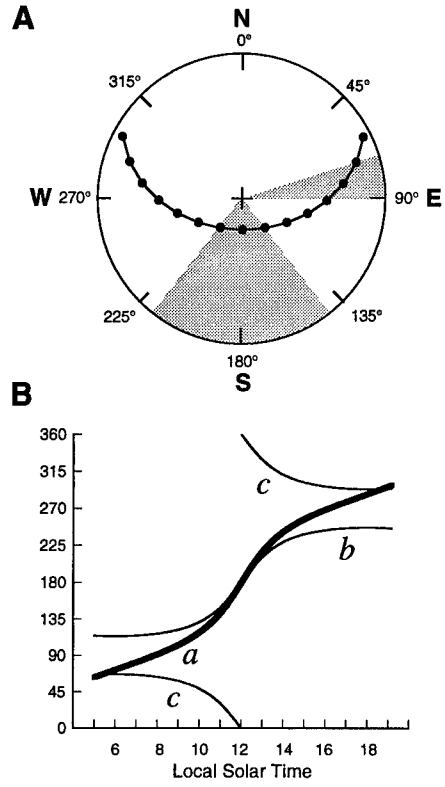


Figure 1. A. The sun's course on the summer solstice in East Lansing, Michigan (≈43° N). Symbols show hourly positions of the sun in the sky from 0500 h to 1900 h. The sun moves at a constant rate (15°/h) along its arc, but the rate of change of the azimuth varies over the day as the sun rises toward the zenith (cross) and then descends again. Shaded sectors show the change in azimuth over two equal (2-hour) periods. The rate of change is 9.5°/h in the morning and 37.5°/h during the period spanning noon. B. Alternative method for displaying change in azimuth over the day (the solar ephemeris). Heavy line (*a*) is plot of sun's course shown in A. Also shown are the ephemerides at the equator on the December solstice (*b*), when the azimuth shifts clockwise (left-to-right) across the southern horizon, and on the June solstice (*c*), when the azimuth shifts counter-clockwise across the northern horizon. (From Dyer and Dickinson in press).

2. Behavioral Evidence

In the original experiment showing that insects could fill gaps in their experience of the sun's course, Lindauer (1959) reared bees in an incubator and allowed them to fly and see the sun only from noon onward each day. During the daily flight period, he trained them to find a feeding station (a dish of sugar water) in the south. After several days of training, he moved the hive overnight to a new terrain (thus depriving the bees of familiar landmarks and forcing them to use the sun as a reference for orienting their flights). He put out an array of baits to determine which direction bees would fly in search of food. Previous work with fully experienced bees had shown that bees search in the compass direction in which they had previously found food, even if they make their trip at a time of day different from the training time, hence with the sun in a different position relative to the line of flight (reviewed by von Frisch 1967). In Lindauer's experiments, the bees were tested in the morning, when they had never seen the sun. They flew predominantly to the south, suggesting that they were informed at least roughly about the morning position of the sun.

Subsequent workers, using other experimental approaches with honey bees (*Apis* spp.), and desert ants (*Cataglyphis* spp.) have replicated Lindauer's general finding that insects can estimate unknown positions of the sun. Perhaps most striking is the evidence that honey bees (*Apis mellifera*, Lindauer 1957; *A. dorsata*, Dyer 1985) and desert ants (Wehner 1982) can estimate the sun's position at night.

Through this work, three specific hypotheses emerged regarding the nature of this estimation. What they share in common is the assumption that the insect calculates unknown positions of the solar azimuth by compensating at a uniform rate (e.g., in degrees/hour) from a known time-linked position of the azimuth. All three hypotheses can be described by the following general expression:

$$A_{\tau+t} = A_\tau + r \bullet t \,, \qquad (1)$$

where $A_{\tau+t}$ is the unknown azimuth, A_τ is the azimuth measured (or previously calculated) at time τ, r is the linear rate of compensation (degrees/time), and t is the time interval between the two determinations of the sun's position.

Where the previous hypotheses differ is in how the animal is assumed to obtain r, the linear compensation rate, from known portions of the sun's course. The models proposed were as follows: 1, linear interpolation between the nearest known time-linked positions of the azimuth (New and New 1962; Wehner 1982); 2, forward extrapolation from the position and rate of movement most recently seen (Gould 1980; Dyer 1985); 3, "backward extrapolation" from the position and rate of movement seen at later times of day (on previous days)(Dyer 1985).

Until recently, it was not clear which of these hypotheses was correct, although the bulk of the evidence supported the linear interpolation model. Recent studies of honey bees (Dyer and Dickinson 1994, in press) and desert ants (Wehner and Müller 1993) suggest, however, that none of the three is correct, and have led to a completely different view of how insects estimate unknown positions of the solar azimuth.

In our experiments (Dyer and Dickinson 1994), we started with Lindauer's (1959) technique of rearing bees in an incubator and allowing them to see only a restricted portion of the sun's course each day. We used the dance language of the bees to infer how they estimated the sun's position at other times of day. The dance language is a code whereby a successful forager communicates the direction and distance of food to her nestmates (von Frisch 1967). The direction coded in the dance is the orientation of the foraging flight relative to the sun (Fig. 2A). Thus, an observer who knows the direction of flight (because the bees are flying to an artificial feeding station in a known location) can infer from the dance where the dancer has determined the sun to be. This technique has been used previously to show that experienced bees draw upon a memory of the sun's course relative to features of the terrain when they have flown under an overcast sky, and hence have not seen the sun during the flight (Dyer 1987). We tested bees on overcast days, so that they were forced to use an internal estimate of the azimuth. By testing them at a time of day when they had never seen the sun, we could determine from their dances how they estimated positions of the sun that they had never seen.

Fig. 2B shows the results from one of our experiments, involving bees that had previously seen the sun only in the late afternoon, and now were flying to a familiar feeding place (under a cloud cover) at other times of day. The three linear compensation models proposed previously do a poor job of accounting for the bees' estimates of the morning and midday course of the sun. The data suggest that the bees, rather than using a single linear rate of compensation during times of day when they have not seen the sun, are roughly informed of overall pattern of solar movement: that the azimuth near sunrise is about 180° from the azimuth near sunset, that the azimuth changes relatively slowly during the morning and during the afternoon, and that the azimuth moves rapidly from one side of the sky to the other at midday.

Our data from restricted-experience bees could be described by a simple 180° step-function, in which a single morning azimuth is replaced by a single afternoon azimuth (different by 180°) at midday. Various evidence (Dyer and Dickinson 1994) suggests that the bees' internal ephemeris is actually a continuous function with a sharp inflection at midday. For example, bees that danced during the midday transition changed their angles continuously (if quickly) in the direction of the afternoon angle.

In another experiment, we gave one group of bees more extensive experience throughout the day during the training phase of the experiment. During the cloudy-day test, these bees tracked the course of the sun more accurately by memory, as in previous experiments (Dyer 1987). We concluded that the learning process entails refining the

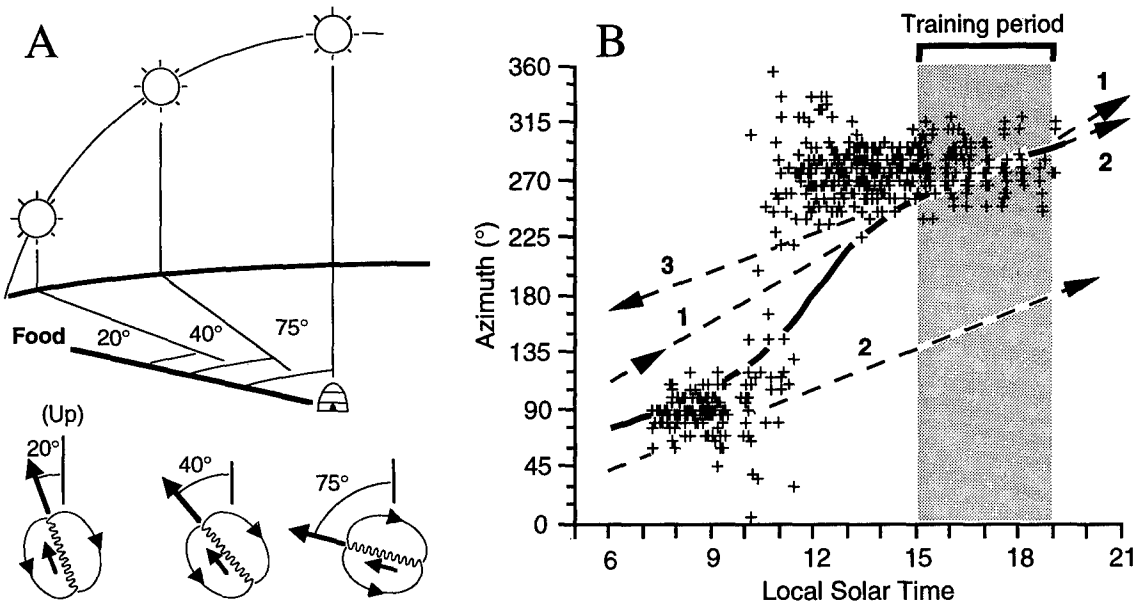


Figure 2. A. Communication of flight direction relative to the sun in waggle dances. The dancing bee repeatedly turns a figure-8 pattern in which the orientation of the straight "waggling run" relative to gravity indicates the angle of the food relative to the solar azimuth. As the azimuth changes during the day, bees dancing to a given feeding site change their orientation by a corresponding amount. B. Solar positions estimated by partially experienced (incubator-reared) bees during an entire cloudy day, as inferred from dances performed in the hive between visits to a feeder. The bees had previously been exposed to a small portion of the sun's course during the late afternoon (shaded area). The cloud cover obscured celestial cues, and hence required bees to remember (or estimate) the sun's position. Each symbol is the solar azimuth inferred for a single dance. Altogether we recorded 554 dances by 46 different bees (this plot omits two bees that used the afternoon azimuth in the morning and the morning azimuth in the afternoon). The curved line shows the solar ephemeris function at the time of this experiment. The dashed lines show the predictions of the three computational mechanisms that had previously been proposed to underlie the ability of bees to estimate unknown positions of the sun (see text): 1, linear interpolation; 2, forward extrapolation; 3, backward extrapolation. (From Dyer and Dickinson 1994, in press)

initial crude sketch of the solar ephemeris function to match the actual ephemeris more closely.

Wehner and Müller (1993), working independently with desert ants (*Cataglyphis fortis*), obtained data that are entirely consistent with our interpretation of the honey bees' behavior. They allowed ants experience outside the nest only in the early morning, then tested their ability to use the sun in the afternoon. The ants behaved as if they expected the sun to be about 180° from its morning position, thus appeared to use an approximation of the sun's course similar to the bees' 180° step-function. With more extensive experience throughout the day, the ants, like the honey bees, estimated the sun's afternoon position more accurately, as if the initial approximation had been refined through learning to correspond to the actual course of the sun.

This mechanism would offer certain advantages to these small-brained, short-lived organisms (see Dyer and Dickinson in press). A naive insect must quickly "discover" the current local solar ephemeris so that it can get on with its main task of collecting food for the colony: honey bee workers spend only about 10 days as foragers before they die, and they make only a few flights from the nest prior to beginning foraging. Using an innate representation of the sun's general pattern of movement could facilitate the process of discovering the actual ephemeris, because the default ephemeris would have to change less to conform to any given actual ephemeris than one that by default estimates the solar azimuth at random at different times of day. Furthermore, an approximate ephemeris in the shape of a 180° step function could be used with relatively little error throughout the day. By using an approximate ephemeris and, as a backup, familiar landmarks to pinpoint the location of the nest (reviews by Dyer 1994, 1996), bees could begin foraging even before they have had the opportunity to refine their representation of the sun's course.

Thus, the recent behavioral evidence has greatly clarified the capacities of the mechanism

whereby bees and ants acquire their knowledge of the sun's course. The data seriously undermine previous hypotheses about the nature of this mechanism, and provide clues that have led to a new hypothesis. These new results leave many questions unanswered, however. Most important for our present purposes, just what computational processes underlie the development of the approximate ephemeris developed by partially experienced insects, and the further refinement of this internal ephemeris as the animals obtain additional experience? The following sections consider the sorts of answers that might be provided by contrasting modeling approaches.

3. A Symbolic Representation

The three hypotheses previously proposed to account for the ability of insects to fill gaps in their experience of the sun's course are symbolic models: all assume that the animal's brain finds unknown positions of the sun by carrying out a calculation on neurally encoded symbols of time and azimuth. The recent behavioral evidence weighs against the specific calculations proposed by these hypotheses. It does not, however, exclude the possibility that the insect's brain performs some other symbolic calculation. Thus we cannot on behavioral grounds reject the view that the sun compass consists of an explicit "equation in the head" that, with appropriate parameters, describes the course of the sun. The question is what sort of equation could account for the behavior. We address this question mainly in reference to the bee data.

Any successful model must at least account for the following properties of the bees' internal ephemeris function: (1) it is cyclic over 24 hours, so that bees expect the sun to rise in the same place each day, and can estimate nocturnal as well as diurnal positions of the sun; (2) it is continuous; (3) it incorporates variations in the rate of change of the azimuth over the day (hence is "non-linear"), and the pattern of these variations is symmetrical around noon; (4) the shape of the function can change to match the actual solar ephemeris observed at different dates and latitudes. Of the various functions that have these properties, the most obvious to consider is the actual trigonometric expression for calculating azimuth as a function of time of day, latitude and solar declination. The difficulty with this function, however, is that insects probably do not have a way to measure their latitude or the current solar declination.

We have explored instead an idea originally suggested to us by Rudolf Jander (personal communication). His suggestion was to describe sun's course as an ellipse centered at the origin of a polar coordinate system. Each point on the ellipse is the rate of change of the azimuth, r (distance of the point from the origin) at a given time of day, τ (the angle relative to coordinate axes). Ellipses of different shapes correspond approximately to differently shaped solar ephemeris functions. For a highly squashed ellipse, r is very large at noon and midnight, and very small at sunrise and sunset; this corresponds to the pattern of solar movement at low latitudes. For a more rounded ellipse, there is less variation in r throughout the day, as at high latitudes. If bees could, by observing a portion of the sun's course, estimate the ellipse corresponding to the current local ephemeris function, then this might provide them a way to estimate the solar azimuth at other times of day. Perhaps bees with relatively little experience by default use a relatively squashed ellipse, which would tend to produce an ephemeris function resembling a 180° step function.

To formalize this hypothesis, we write the equation of the ellipse in polar coordinates as:

$$r = \frac{a \bullet b}{\sqrt{a^2 \sin^2\tau + b^2 \cos^2\tau}}, \qquad (2)$$

where a and b are the principle axes of the ellipse, and indicate whether its shape is squashed ($a \gg b$) or rounded ($a \approx b$). To find the ellipse corresponding to the current ephemeris function, the parameters a and b can be estimated from measurements of the rate of change of the azimuth at two or more different times during the day (Dickinson and Dyer in prep.). To calculate the azimuth at other times of day, Eq. 1 can be implemented as a difference equation, with Eq. 2 providing the parameter r for each time interval. The difference equation calculates positions of the solar azimuth (A_τ) after successive small time intervals (t) starting at some known azimuth. Changing the sign of the equation would produce an ephemeris function in which the azimuth goes counter-clockwise rather than clockwise, corresponding to the situation at southern latitudes (Fig. 1).

This method can produce ephemeris functions that closely matches the actual ephemeris function at both high and low latitudes (Fig. 3). Thus, we have shown that a single general function,

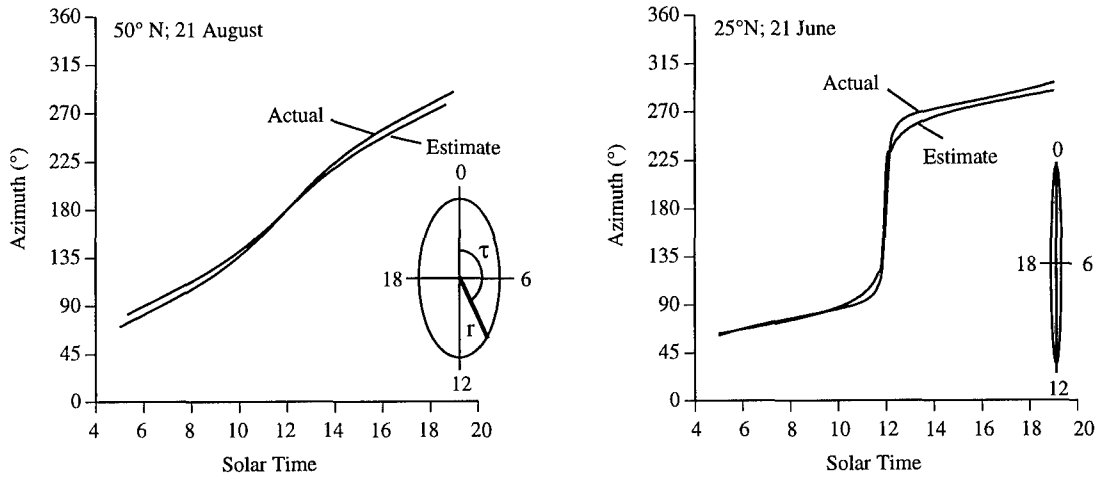


Fig. 3. Symbolic model of sun-learning that uses the equation of an ellipse in polar coordinates (Eq. 2 in text) as the measure of the rate of change of the azimuth. Inset in each figure is the ellipse estimated from measurements of the sun during a restricted portion of the day. See text for definitions and further explanation.

operating on information potentially available to insects, can produce a family of estimated ephemeris functions that roughly resembles the family of actual ephemeris functions. The performance of this symbolic model matches the insects' behavior in ways that the previous linear computational models (Fig. 2B) could not do. It is of course an entirely different matter whether this model describes what goes on in the bee's brain.

4. Connectionist Representations

In the approach that we have just described, the internal representation of an environmental pattern is modeled as an explicit (if approximate) mathematical function operating on neural symbols of external events. For many years, this "symbolic paradigm" (Smolensky 1988) dominated research in cognitive psychology and ethology (for recent applications to problems in spatial orientation, see Schöne 1984 and Gallistel 1990). The implication of much of this work is that neural processes in the brains of animals actually implement such mathematical functions, although the goal of many researchers is to develop functionalist theories of cognitive processes that are independent of a particular neurobiological implementation.

Connectionist models avoid the assumption that neural processes implement symbol-manipulating mathematical functions. Instead, as mentioned earlier, environmental patterns are represented in patterns of activation distributed across highly interconnected networks of neuron-like units. Connectionist models have been developed that mimic numerous phenomena of perception, learning and action (see Churchland 1995 for a recent general review). Could a suitably configured neural network mimic the performance of honey bees in learning the solar ephemeris function? Here we consider two distinct connectionist architectures, one more plausible than the other.

4.1 A Perceptron Model

As a first step, we conceived of the sun-learning problem as a case of perceptual recognition, a task for which connectionist models known as "perceptrons" have proven particularly well-suited. The task that a partially-experienced bee faces is to "recognize" the overall pattern of solar movement even if she has only been exposed to a portion of the actual sun's course. This is loosely analogous to the task faced by a neural network that has been trained to recognize individual faces, and is then confronted with incomplete or garbled images of these faces (Cottrell 1991). In connectionist networks that can solve this problem, the test pattern elicits an activation vector (which in turn triggers a particular output corresponding to "recognition") that resembles the activation vector elicited by a pattern familiar to the network. The network thus can "fill gaps" in the test pattern in the sense that a familiar activation vector is produced in spite of such gaps. Could the ability of bees and ants to fill gaps in their experience of the sun's course be explained in this way?

A common way of designing a connectionist neural network to handle such recognition

problems is to expose it repeatedly to a set of examples of the patterns to be recognized, while incrementally adjusting connections between units in the network until it produces the correct output for each of the training patterns. With a neural network simulation package (Caudill and Butler 1992), we used this sort of supervised training process on the three-layer feed-forward network shown in Fig. 4A. The input layer consists of an array of units sensitive to specific combinations of visual-spatial (azimuth) and temporal (time-of-day) inputs. Each unit in the input layer connects to all of the units in a hidden layer of 25 units. The units in the hidden layer connect in turn to an output layer in which the estimate of the solar ephemeris function is generated and fed to the orientational control systems. The topology of the output layer is identical to that of the input layer.

We exposed this network to a simplified family of 7 sun-azimuth functions representing a range of latitudes and seasons (Fig. 4B). Through back-propogation of error, we initially trained the network to produce in the output layer a pattern identical to the pattern in the input layer. Fig. 4A shows the activation levels in the hidden and output layers of a trained network when probed with one of the training patterns.

After training the network, we tested it by exposing it to various fragments of a solar ephemeris function (Fig. 4C). We wondered whether the network would mimic the ability of partially experienced insects to estimate the sun's course at times of day when they have not seen it.

We found that the network did indeed mimic the insects' behavior in various ways. Most important, it tended to produce an estimate resembling a 180° step function, and it did so whether the fragment to which it was exposed fell in the late afternoon (resembling what our honey bees experienced) or the early morning (as Wehner and Müller's desert ants experienced). In addition, for one of the late-afternoon fragments, the network produced a pattern that resembled a step-function, but appeared to exhibit more variability during the midday transition. This result mimics the data in Fig. 2B, which we interpreted as evidence that bees disagreed, or were confused, about which way to change their orientation to span the 180° gap from the morning azimuth to the afternoon azimuth (Dyer and Dickinson 1994).

Although the model mimics certain aspects of insect behavior, two facts about sun-learning by

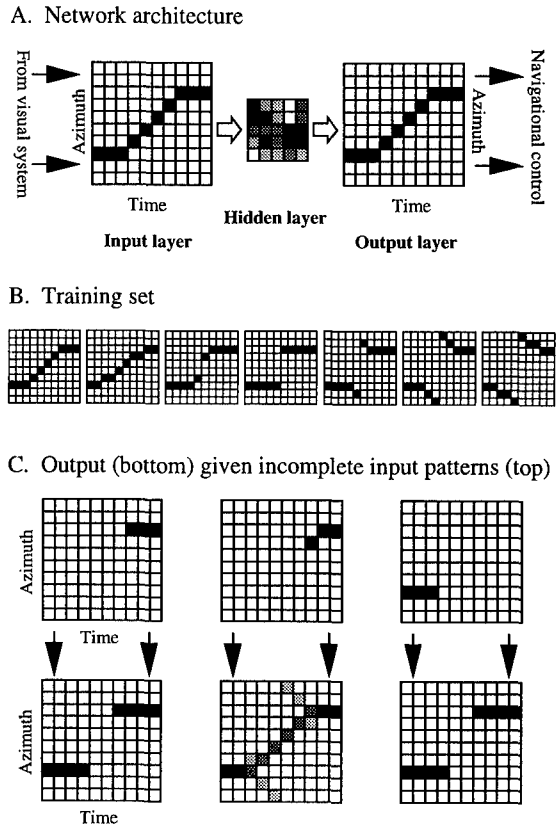


Fig. 4. Performance of a connectionist model of sun-learning based on a three-layer, feed-forward perceptron architecture trained with back-propagation of error. In each diagram, shading gives the relative degree of activation of each unit in the network. See text for further explanation.

insects expose limitations in the plausibility of this model. First, remember that individual bees and ants can fill in unknown portions of the sun's course even though they have not been exposed to examples of the entire sun's course. Thus, the insect apparently begins life with a neural network that is already configured to solve the filling-in problem that we are posing to the artificial network. To apply this model to insects, we must assume that the training of their sun-compass network must have happened over evolutionary time, through exposure of the bees' ancestors to examples of sun-azimuth curves, and cumulative selection of the genetically determined initialized state of the network.

Second, and far more serious, bees and ants can estimate the sun's course at night, a time during the daily cycle when neither living insects nor any of their ancestors could have seen examples of the pattern of solar movement. Thus, insects behave as if their "input" and "output" layers differ from those in Fig. 4 in spanning 24 hours and 360° of azimuth, and as if the output layer

can produce estimates of azimuth at times when the input layer has never "seen" the sun. It is hard to imagine what sort of learning or evolutionary process could produce this behavior in a network with the general architecture of the one in Fig. 4. This appears to be a fatal limitation of the model, and of any model that requires exposure to examples of complete patterns to be able to recognize incomplete patterns.

Is a connectionist approach therefore doomed to failure as route to understanding the insect's internal ephemeris function? Not necessarily. We can rescue the connectionist approach either through ad hoc modifications of the generalized perceptron architecture depicted in Fig. 4, or by designing a network whose architecture is purpose- built for the task of representing the sun's complete course.

We consider the first solution only briefly. The principal failing of the model in Fig. 4 is that it cannot through individual experience or through selection over evolutionary time be trained to estimate nocturnal positions of the azimuth. One way to solve this problem would be to construct a "copy" of network so that the diurnal activation pattern can be activated at night. This would exploit the resemblance between the nocturnal and diurnal courses of the sun: the nocturnal "azimuth" shifts relatively slowly at times far from midnight (just after sunset and just before sunrise), and relatively quickly near midnight, when the sun is opposite the local meridian. Thus, the azimuth estimated 3 hours after sunset (relative to the azimuth at sunset) would usually mirror the azimuth estimated 3 hours after sunrise (relative to the azimuth at sunrise).

Such ad hoc modifications add little to the plausibility of the model in Fig. 4, which already lacks realism in various ways. We believe that it is therefore more fruitful to consider connectionist architectures that are better suited to the task.

4.2 Cyclic Neuronal Chains

In an alternative approach (Dyer and Dickinson in preparation), we have considered a connectionist architecture based on cyclic chains of neural units. Cyclic neuronal chains have proven very useful in modeling spatial learning phenomena (e.g., Hartmann and Wehner 1995; McNaughton et al. 1996). They seem especially appropriate in the present situation because the pattern to be represented is composed of three cyclic patterns: time of day, the 360° cycle of solar movement, and the 360° panorama of landmarks around the nest, which is used as the reference for learning the sun's course (Dyer 1987).

The core of our model consists of two interconnected layers. One layer, which receives inputs from the visual system, maps possible positions of the solar azimuth over 360° (as referenced to the panorama of landmarks around the nest). The other layer, which receives input from the internal clock, is essentially a map of time of day, such that different units are maximally active at different times of day.

We assume that each Time unit sends excitatory connections of variable strength to every Azimuth unit. Such a network can then represent any solar ephemeris function by having some Time-Azimuth connections strengthened relative to others. Fig. 5 shows how, with different configurations of connection weights, the same basic architecture can represent the gradual ephemeris function at Lat. 50° in late August, and the sharply changing ephemeris function at Lat. 25° on the June Solstice. The counter-clockwise rotations of the sun's azimuth in the southern hemisphere could also be accommodated by this architecture.

The situation gets a little more complicated when we try to imagine what learning rule could produce the different configurations of the network shown in Fig. 5. One possibility is a Hebbian process whereby the connection between a specific Time unit and a specific Azimuth unit is strengthened relative to other connections if they are simultaneously active. Thus, the Time unit active at 1400 h would connect more strongly to the Azimuth unit that was stimulated at that time than to other Azimuth units. As the animal accumulates experience over the day, different Time units would tend to become strongly connected to different Azimuth units, at least when confronted with a gradual ephemeris function.

There is a major limitation of this simple mechanism, however: the best that it could do is to produce the equivalent of a lookup table encoding time-linked solar positions actually observed. Further features must be added for the network to estimate the sun's position at times of day or night when the animal has never seen it. For example, as we discuss in detail elsewhere (Dickinson and Dyer in press), there at least have to be excitatory connections within the Time layer and within the Azimuth layer, such that an active unit within each layer tends to produce activation in other units. The strongest connections have to be between units on opposite sides of the cyclic chain (e.g. between Time units τ and τ+12 h, and between Azimuth

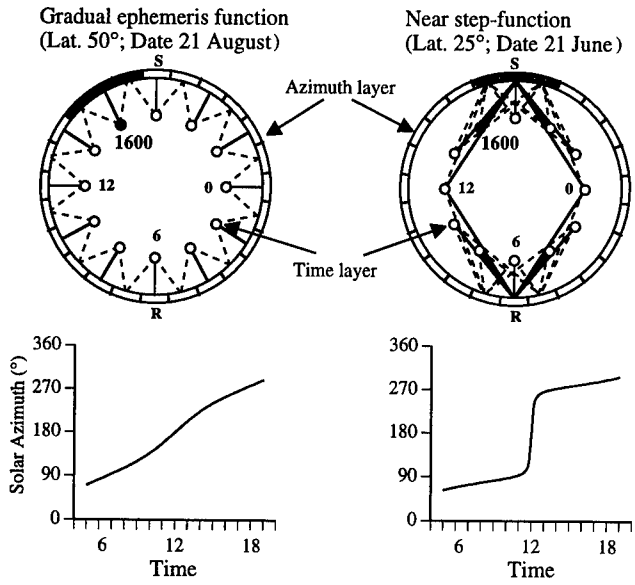


Fig. 5. Performance of a connectionist model of sun-learning based on two layers, each consisting a unidimensional neural chain with cyclic topology. The Azimuth layer maps possible positions of the solar azimuth over 360° (R and S indicate the rising and setting positions of the sun). The Time layer maps time of day. Each Time unit connects with all Azimuth units. With connections strengthened selectively (shown by lines connecting Time layer to Azimuth layer), activity in a given Time unit will tend to activivate a specific Azimuth unit. The connection patterns in the network as a whole can represent any ephemeris function. Shown here are connection patterns for two different ephemeris functions, and the activity pattern produced in the Azimuth layer at 1400 h. Each network represents the course of the sun over 24 h (the graphs show only the diurnal portion of the sun's course).

units A and A+180°). In this way, Hebbian reinforcement of the connection between the currently most active Time and Azimuth units (units τ and A) would be accompanied by reinforcement of the connection between units τ+12 and A+180°. This would partially fill gaps in the network's experience (other connections must be introduced for the gaps to be filled completely), including nocturnal portions of the sun's course. Furthermore, the representation formed by a partially experienced network would resemble the bee's innate ephemeris in being symmetrical and describable by a 180° step function.

Before leaving this discussion of connectionist approaches, we would like to highlight an important question that has emerged in our exploration of connectionist models of the sun-learning process: how is time of day encoded and used by the brain? Gallistel (in press) argues that the ability to use time of day to update an estimate of the azimuth is evidence that the animal's brain uses the output of the clock as a "symbol" in the computation that generates the estimate. In the connectionist models that we have examined, the output of the clock jointly determines (with visual measurements of solar position) which elements of a neural array contribute to a particular activation vector in the network. The output of the clock could be viewed as a symbol in the semantic sense that it refers to a particular environmental condition (time of day), but it is not a symbol in the syntactic sense that it is used in a symbol-processing mechanism that computes azimuth (see Smolensky 1988).

A further point to make regarding the role of time is that both of our connectionist models assume that time of day is physically mapped into the network. Thus, different units are assumed to be maximally active at different times of day. To our knowledge, there is so far no evidence that time of day is physically mapped into real nervous systems in this way. It is not immediately obvious, however, how a connectionist approach could be made to work without such a physical mapping of time of day.

We conclude that Gallistel (in press) is partially justified in his skepticism about the ability of connectionist neural network models to account for the phenomena of sun-learning in insects. Many general-purpose perceptron architectures that are otherwise capable of sophisticated tasks of pattern recognition would be incapable of producing estimates of the sun's course at night as insects and other animals can.

As we have shown, however, a perceptron architecture can reproduce some of the behavioral phenomena observed in insects, however, including the basic ability to keep track of the diurnal course of the sun. Furthermore, a more specialized model, its architecture designed to match the parameters of the learning problem, can reproduce the animal's behavior very convincingly, without encoding an explicit mathematical description of the sun's course. This at least moves us part of the way toward an answer to Gallistel's question: "What does a non-symbolic ephemeris function look like?"

5. General Conclusions

Our aim in this paper is not to defend a particular approach to modeling cognitive processes, but to examine the sorts of assumptions that are required to make each approach work as an account of the

phenomena of sun-learning in insects. We regard the connectionist paradigm as somewhat more plausible than the symbolic paradigm, although we recognize that the connectionist models that we have explored suffer acutely from a lack of evidence regarding how realistic their assumptions are.

Future attempts to understand the processes underlying sun-learning could lead to new insights into the behavior of animals as well as robots. First, sun-learning may involve representational processes different from those involved in the representation of spatial relationships among landmarks, and thus further studies are likely to broaden our understanding of the principles by which nervous systems represent environmental patterns. One outcome of such studies may be to provide insights into how time of day is used in learning processes.

Second, studies of sun-learning provide an opportunity to examine how information that an individual acquires through its own experience can be integrated with innately specified (evolved) information.

Third, the correspondence between the insect's innate ephemeris and the general pattern of solar movement may be a useful design principle for a representational system that can expect certain regularities in the pattern to be learned (Dyer and Dickinson in press). Honey bees that have sampled only a small portion of the sun's course can estimate its relative position at other times of day with relatively little error--certainly far less error than if they estimated the azimuth at random. This may reduce the need to take valuable time out of their short lives to sample the sun's course extensively before they begin to search for food. Furthermore, having an innate ephemeris that is already similar to any actual ephemeris could speed up the process of developing a closer match between the internal ephemeris and the current local ephemeris. Conceivably, this design principle, if that is what it is, could be incorporated into robots that need to solve tasks in environments in which the parameters of the tasks can be partially but not completely specified.

Acknowledgements

We are grateful to Rudolf Jander for advice and discussion, and to the National Science Foundation and Michigan State University for supporting the research.

References

Caudill, M., and Butler, C. (1992). Understanding Neural Networks: Computer Explorations. Cambridge, MA: MIT Press.

Churchland, P. M. (1995). Engine of Reason, Seat of the Soul. Cambridge, MA: MIT Press.

Cottrell, G. (1991). Extracting features from faces using compression networks: Face, identity, emotions and gender recognition using holons. In: Connectionist Models: Proceedings of the 1990 Summer School. (ed. Touretzky, D., et al.). San Mateo, CA: Morgan Kaufmann.

Dickinson, J. A., and Dyer, F. C. (in prep.) Models of sun-azimuth learning in insects.

Dyer, F. C. (1985). Nocturnal orientation by the Asian honey bee, Apis dorsata. Anim. Behav. 33, 769-774.

Dyer, F. C. (1987). Memory and sun compensation in honey bees. J. Comp. Physiol. A 160, 621-633.

Dyer, F. C. (1994). Spatial cognition and navigation in insects. In Behavioral Mechanisms in Evolutionary Ecology (ed. L. A. Real), pp. 66-98. Chicago: University of Chicago Press.

Dyer, F.C. (1996). Spatial memory and navigation by honey bees on the scale of the foraging range. J. Exp. Biol. 199, 147-154.

Dyer, F. C., and Dickinson, J. A. (1994). Development of sun compensation by honey bees: How partially experienced bees estimate the sun's course. Proc. Natn. Acad. Sci. U.S.A. 91, 4471-4474.

Dyer, F. C., and Dickinson, J. A. (in press). Sun compass learning in insects: representation in a simple mind. Curr. Dir. Psychol. Sci.

Frisch, K. v. (1967). The Dance Language and Orientation of Bees. Cambridge, MA: Harvard University Press.

Gallistel, C. R. (1990). The Organization of Learning. Cambridge, MA: MIT Press.

Gallistel, C. R. (in press). Brains as symbol-processors: the case of insect navigation. In: An Invitation to Cognitive Science, Volume 4: Conceptual and Methodological Foundations (ed. S. Sternberg & D. Osherson). Cambridge, MA: MIT Press.

Gould, J. L. (1980). Sun compensation by bees. Science 207, 545-547.

Hartmann, G., and Wehner, R. (1995). The ant's path integration system -- A neural architecture. Biol. Cybern. 73, 483-497.

Lindauer, M. (1957). Sonnenorientierung der Bienen unter der Aequatorsonne und zur Nachtzeit. Naturwissenschaften 44, 1-6.

Lindauer, M. (1959). Angeborene und erlernte Komponenten in der Sonnenorientierung der Bienen. Z. vergl. Physiol. 42, 43-62.

McNaughton, B., Barnes, C. A., Gerrard, J. L., Gothard, K., Jung, M. W., Knierem, J. J., Kudrimoti, H., Qin, Y., Skaggs, W. E., Suster, M., and Weaver, K. L. (1996). Diciphering the hippocampal polyglot: the hippocampus as a path integration system. J. Exp. Biol. 199, 173-185.

Müller, M., and Wehner, R. (1988). Path integration in desert ants, Cataglyphis fortis. Proc. Natn. Acad. Sci. U.S.A. 85, 5287-5290.

New, D. A. T., and New, J. K. (1962). The dances of honey-bees at small zenith distances of the sun. J. exp. Biol. 39, 279-291.

Schöne, H. (1984). Spatial orientation. Princeton, NJ: Princeton University Press.

Smolensky, P. (1988). On the proper treatment of connectionism. Behav. Brain Sci. 11, 1-23.

Wehner, R. (1982). Himmelsnavigation bei Insekten. Neurophysiologie und Verhalten. Vierteljahrsschr. Naturforsch. Ges. Zürich 5, 1-132.

Wehner, R. (1984). Astronavigation in insects. Ann. Rev. Ent.

Wehner, R., and Müller, M. (1993). How do ants acquire their celestial ephemeris function? Naturwissenschaften 80, 331-333.

Spatial exploration, map learning, and self-positioning with MonaLysa

Jean-Yves Donnart and Jean-Arcady Meyer
AnimatLab
Ecole Normale Supérieure
46, rue d'Ulm 75230 Paris Cedex 05, France
(donnart@wotan.ens.fr - meyer@wotan.ens.fr)

Abstract

This paper describes how the MonaLysa control architecture implements a route-following navigation strategy. Two procedures that allow map building and self-positioning are described, and experimental results are provided that demonstrate that such procedures are robust with respect to noise. This approach is compared to others with similar objectives, and directions for future work are outlined.

1 Introduction

In robotics or animat research, traditional navigation methods that use internal geometrical representations of the environment (Latombe, 1991) are confronted with various implementation difficulties, due to memory and time requirements, as well as sensory and motor errors (Nehmzow, 1995). To overcome these difficulties, several researchers (Chatila and Laumond, 1985; Kuipers and Byun, 1991; Mataric, 1992; Nehmzow, 1995) have advocated the use of various types of topological models to represent the connectivity of the environment, and several such models have been devised that aim to mimic known nervous architectures or behavioral capacities in animals (Muller et al., 1991; Mataric, 1992; Schmajuk and Thieme, 1992; Penna and Wu, 1993; Bachelder and Waxman, 1994; Nehmzow, 1995; Scholkopf and Mallot, 1995). Basically, these topological models endow an animat or a robot with cognitive abilities that make possible to " recognize " the place it is situated in and to " know " that, if it performs a given move in a given direction, it will arrive in another " known " place. In other words, such models belong to the category of so-called *world models*, and they encode a variety of *Stimulus-Response-Stimulus* (S-R-S) information (Riolo, 1991; Roitblat, 1994). At the functional level, they allow the animat or the robot to navigate according to a *route-following* strategy (Gallistel, 1990) and to plan a trajectory from a given starting place to a given goal place, provided that such a trajectory can pass through already known places and involves already experienced moves from place to place. They also facilitate place recognition because such a task may take into account, not only the various features that characterize a given place, but also the specific moves that lead to that place, or depart from it, together with the specific places that are thus connected to it. Moreover, additional information can also be taken into account and facilitate the place-recognition task. This is, for instance, the case with Mataric's robot (Mataric, 1992), that uses the metric information - like distances and orientations - provided by specialized sensors. This is also the case with Kuipers and Byun's animat (Kuipers and Byun, 1991) that uses the information provided by the control architecture that governs its successive moves.

This paper is primarily concerned with the place-recognition and self-positioning functionalities. It describes how it has been possible to add to the basic exploratory abilities of an animat endowed with the MonaLysa control architecture (Donnart and Meyer, 1994; Donnart and Meyer, 1996) the faculties of building a so-called *cognitive map* (Tolman, 1948; O'Keefe and Nadel, 1978; Kuipers, 1982; Thinus-Blanc, 1988; Gallistel, 1990; Poucet, 1993) of its environment, and of locating itself despite the lack of precision in its sensors and actuators. It is structured as follows: Section 2 summarizes the characteristics of MonaLysa that have already been described elsewhere. Section 3 describes how it detects *landmarks* and builds a spatial representation of the environment. Section 4 describes the *self-positioning* procedure. Experimental results are shown in Section 5. The following section discusses the advantages and drawbacks of this approach and ends with perspectives for future work.

2 The MonaLysa architecture.

In the present application, the MonaLysa architecture enables an animat to explore a two-dimensional environment and to navigate from one place to another despite the various obstacles that the environment may contain. The animat is equipped with proximate sensors that keep it informed of the presence or absence of any obstacle in front of it, 90° to its right, or 90° to its left. It is also able to estimate the direction of a goal to be reached in each

of the eight sectors of the space surrounding it. Lastly, it is capable of moving straight ahead, 90° to its right, or 90° to its left. This architecture, which relies upon a hierarchical *classifier system*, is organized into 8 modules - a reactive module, a planning module, a context manager, an auto-analysis module, an internal reinforcement module, a mapping module, a spatial information processor and a place recognition module (Figure 1).

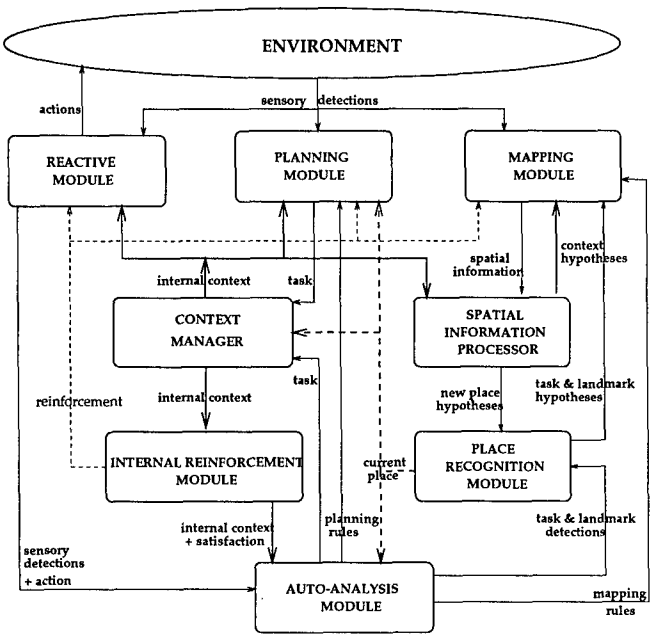


Figure 1: The MonaLysa architecture

The functionalities of the first five modules have been extensively described elsewhere (Donnart and Meyer, 1994; Donnart and Meyer, 1996) and will accordingly be dealt with only briefly. As a whole, they manage a pile of *tasks* that allow the animat to reactively escape from any obstacle it gets trapped into by skirting around it, and to plan a trajectory that will later allow it to avoid the obstacle from a distance. Such an ability relies upon the animat's capacity to analyze its skirting paths and to detect landmarks that will be used for locating itself in the environment.

The reactive module chooses the next move to perform. It uses production rules that take the form :
If <sensory information> and <direction of current goal> Then <action>

The sensory information that these rules take into account is that provided by the animat's proximate sensors.

The planning module decomposes a task into a series of subtasks, that is, into a series of planned trajectories connecting a starting place to an end place. It uses production rules that take the form :
If <sensory information> and <current task> Then <subtask>

The sensory information that these rules take into account is that provided by the animat's proximate sensors, augmented with the animat's coordinates and current orientation.

The context manager contains a pile of the tasks that the animat autonomously generates while it moves in its environment. Such tasks can be either *skirting tasks*, that are posted by the auto-analysis module when the animat detects an obstacle, or *obstacle-avoidance tasks* or subtasks, that are posted by the planning module. At each instant, the task at the top of the pile specifies the *current goal* and the *current task*, i.e., the *internal context* in which reactive and planning rules can be triggered.

The internal-reinforcement module is used to monitor the *satisfaction* of the animat and to adjust the *strengths* of the reactive rules. The satisfaction is an estimation of the success with which a given rule brought the animat closer to, or took it farther from, its current goal. The strengths are used to probabilistically choose which rule to trigger if the condition parts of several rules match the current situation

The role of the auto-analysis module is to analyze the current behavior of the animat in order to alter its current task dynamically and to create new tasks that will enhance its future behavior. It is also responsible for characterizing landmarks in the environment, i.e. places through which it will be useful to travel in the future in order to avoid the obstacles from a distance, or which are used for map building and place recognition.

When an obstacle is detected, as described in (Donnart and Meyer, 1996), the auto-analysis module generates a skirting task that specifies that the animat must cross the line that lies parallel to the direction taken to avoid the obstacle, and that passes through the place where the obstacle has been detected. This task is coded by the pair *<coordinates of the place> < direction vector of the straight line to be crossed>* and is registered at the top of the pile of the context manager. An emergent functionality of the internal dynamics of this module is to enable the animat to skirt around the obstacles it encounters and to extricate itself from dead-ends with arbitrarily complicated shapes.

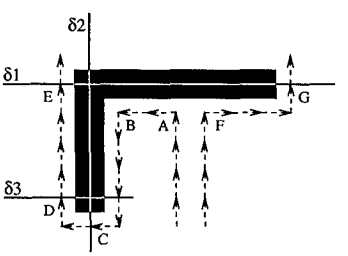


Figure 2: Skirting tasks and skirting trajectories generated by MonaLysa.

For example, two skirting trajectories around a specific obstacle are shown on Figure 2. The left trajectory has been generated through the creation of three skirting tasks at places A, B and C, respectively associated with lines $\delta 1$, $\delta 2$ and $\delta 3$. The task associated with line $\delta 2$ - that will simply be called task $\delta 2$ from now on - has been erased at place C, and tasks $\delta 3$ and $\delta 1$ have been erased respectively at places D and E. The right trajectory has been generated through the creation of task $\delta 1$ at place F and its erasing at place G.

3 Landmark detection and map learning

Another role of the auto-analysis module is to analyze the trajectory followed by the animat in order to detect landmarks in the environment. There are 4 categories of such landmarks : *satisfying* places, *unsatisfying* places, *task-erasing* places and *marker* places.

To detect landmarks belonging to the first two categories, the auto-analysis module monitors the variation of the animat's satisfaction between two successive moves. It also detects *constrained moves*, that is, moves that aren't the most conducive to reaching the goal, and to which the animat resorts because of the presence of an obstacle. Such moves trigger obstacle detection, as described in (Donnart and Meyer, 1996). Places where the satisfaction gradient is positive (resp. negative) and that have been reached through a constrained move are categorized as satisfying (resp. unsatisfying) landmarks. Some such satisfying landmarks are used to define the start and end places of the obstacle avoidance subtasks generated by the planning module, according to a recursive procedure that has also been described in (Donnart and Meyer, 1996). Task-erasing landmarks are places where a task can be erased from the context manager pile. Finally, marker landmarks are places that belong to any of the three preceding categories. They are defined as any first landmark the animat is capable of encountering after the triggering of a skirting task, or as any last landmark the animat is capable of encountering in the context of this skirting task. It turns out that such landmarks are useful for expediting the hierarchical positioning procedure described below.

The " map " that the animat builds while it explores its environment has nothing in common with a classical two-dimensional map. Rather, it is coded as various production rules that are created by the auto-analysis module, that are memorized into the mapping module, and that post their action part to the spatial information processor. Such rules belong to 6 different categories and take the forms :

(1) *If <marker landmark A> and <current task δ > Then <record the pair (A, δ)>*

(2) *If <marker landmark A> and <current task $\delta 2/\delta 1$> Then <record the pair (A, $\delta 2$)>*

(3) *If <link between two landmarks (A, B) detected> and <(X, δ) recorded> Then <support structure [X, Y]/δ >*

(4) *If <unsatisfying landmark A and subtask $\delta 2$ detected> and <(X, $\delta 1$) recorded> Then <support structure [X, Y]/$\delta 1$>*

(5) *If <marker landmark Y> and <(X, δ) recorded> Then <record the structure [X, Y]/δ >*

(6) *If <task-erasing landmark A and new task $\delta 2$ detected> and <current task $\delta 1$ to be erased> Then <record the pair (A, $\delta 2$)>*

where expressions like <task $\delta 2/\delta 1$> and <structure [X, Y]/$\delta 1$> mean that task $\delta 2$ and structure [X, Y] are detected within the context of a specific task $\delta 1$.

Thus, such rules are capable of recording the association between a landmark and a skirting task (type 1), or the association between a landmark and two skirting tasks, which can be hierarchically (type 2) or sequentially (type 6) linked. Such rules are also capable of recording more global *structures* that include two or more landmarks and subtasks (type 5). Finally, they are capable of providing support to some position hypotheses (types 3 and 4), as explained later.

When they involve a skirting task, the corresponding records take into account the coordinates of the place where the task is triggered and the direction vector of the straight line to be crossed. A position and an orientation error are also recorded. When they involve a landmark, the corresponding records take into account the coordinates of the corresponding place and the orientation of the animat in this place. Such records also take into account on-line estimates of the animat's position and orientation errors and the category to which the detected landmark belongs. It must be noted that, although *absolute* coordinates and orientations are recorded into the mapping rules, the matching process between two places or two tasks that will be described later involves *relative* distances or orientations between such places and tasks. Absolute information arbitrarily refer to the animat's initial position. It is taken into account only when matching is ambiguous - for instance when a given place can be matched to two, or more, other places - or when two places are too far apart to be linked within any mapping rule - for instance when they belong to two distinct objects in the environment. It must also be noted that no sensory information is used in the description of the landmarks, even if such information would certainly increase the animat's self-positioning capabilities. Therefore, the self-positioning capacities of MonaLysa do not

depend upon the animat's sensory errors, contrary to what happens in many other realizations.

The process of landmark detection and map learning also entails the management of two lists within the auto-analysis module, a LC_list and a S_list, which are associated with each skirting task triggered. The LC_list records each landmark and each skirting subtask that are encountered or generated within the context of a given skirting task. The S_list records specific portions of the animat's trajectory that are traveled within the context of a given skirting task, that begin and end with a marker landmark, and that can be encapsulated as hierarchical structures likely to be used by mapping rules of types 3, 4 and 5.

The management of these lists and the creation of the mapping rules will be illustrated on the specific example of Figure 3. While the animat is traveling southwards, it encounters an obstacle at place A and triggers 3 skirting tasks successively, characterized by lines $\delta 1$, $\delta 2$ and $\delta 3$. Having moved in turn through places A, B, ... I, it resumes traveling southwards from place I.

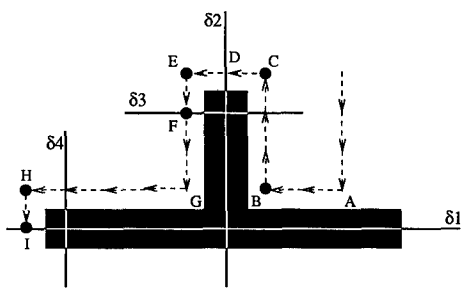


Figure 3: Skirting tasks (δ lines) created and landmarks (filled circles) detected along a trajectory

Along its path, the animat's first encounter with the obstacle occurs at place A. Because the animat cannot proceed straight on, its gradient of satisfaction becomes negative at A. However, this place is not characterized as an unsatisfying place because the move that led to A was not constrained. In place A, the animat chooses to turn right and generates a skirting task characterized by line $\delta 1$. While it moves from A to B, its current goal is to reach the projection of its current position onto line $\delta 1$: the corresponding satisfaction gradient remains constant and no landmark is detected along the path (Figure 4a). However, the corresponding moves are constrained because the presence of the obstacle prevents the animat from turning left, i.e. from reaching its current goal on line $\delta 1$.

In B, the animat encounters a new obstacle, triggers a new task $\delta 2$, and turns to the right. Because its satisfaction gradient with respect to line $\delta 1$ diminishes, and because the preceding move was constrained, place B is detected as both an unsatisfying landmark and a marker landmark within the context of task $\delta 1$. B and $\delta 2$ are added to the LC_list associated with $\delta 1$. Likewise, B is detected as a satisfying landmark and a marker landmark within the context of task $\delta 2$. Thus B is also added to the LC_list associated with $\delta 2$ (Figure 4b). From B to C, the animat's current goal is to cross line $\delta 2$, the corresponding moves are constrained, and the satisfaction gradient remains unchanged.

In C, the animat can turn towards its current goal on line $\delta 2$ and its satisfaction gradient increases : therefore, place C is detected as both a satisfying landmark and a marker landmark associated with $\delta 2$. Place C is then added to the LC_list associated with $\delta 2$ but, because this list begins and ends with two marker landmarks, B and C, these landmarks are encapsulated as a structure [BC] which is added to the S_list associated with $\delta 2$. The corresponding LC_list shrinks into place C (Figure 4c).

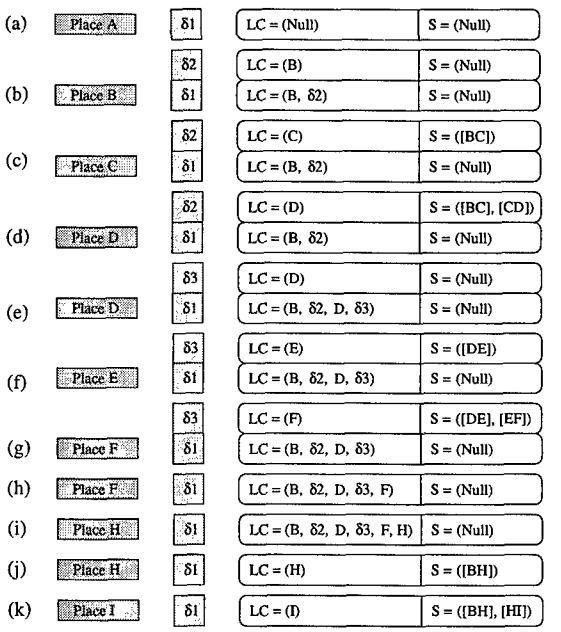


Figure 4: The management of the LC_list and S_list during map learning

When the animat arrives at place D, task $\delta 2$ is erased from the context-manager pile and place D is detected as a task-erasing landmark and a marker landmark that is added to the LC_list associated with $\delta 2$. However, because, this list begins and ends with two marker landmarks, C and D, these landmarks are encapsulated as a structure [CD] which is added to the S_list associated with $\delta 2$. The corresponding LC_list shrinks into place D (Figure 4d). Furthermore, the erasing of task $\delta 2$ allows the creation of 2 type 1 mapping rules, which record the pairs (B, $\delta 2$) and (C, $\delta 2$) and of 2 type 5 mapping rules, which record the structures [B, C]/$\delta 2$ and [C, D]/$\delta 2$. Then, the two lists associated with $\delta 2$ are erased. How-

ever, because the obstacle still prevents the animat from crossing line $\delta 1$, D is also detected as an unsatisfying landmark associated with $\delta 1$ and the skirting task $\delta 3$ is added to $\delta 1$ on the top of the context-manager pile. Finally, within context $\delta 3$, D is also detected as a satisfying landmark and a marker landmark (Figure 4e).

In place E, the animat can turn to its right and move towards its current goal on line $\delta 3$. As the preceeding move was constrained, place E is detected as both a satisfying landmark and a marker landmark associated with $\delta 3$. Place E is added to D, within the corresponding LC_list. This list shrinks to E when structure [D,E] is created and added to the corresponding S_list (Figure 4f).

The same events as those that occurred in place D now occur in place F, which is detected as a task-erasing landmark and a marker landmark in context $\delta 3$, and as a satisfying landmark in context $\delta 1$. In particular, the type 1 and type 5 mapping rules that record pairs (D, $\delta 3$) and (E, $\delta 3$), and structures [C, D]/$\delta 3$ and [D, E]/$\delta 3$ are created (Figure 4g). Then, task $\delta 3$ is erased and the LC_list and S_list of task $\delta 1$ become as shown on Figure 4h.

Place G is not detected as a landmark, because the move that led to it was not constrained. Conversely, place H is detected as both a satisfying and a marker landmark associated with $\delta 1$ and is added to the corresponding LC_list (Figure 4i). Then, the list shrinks to H, while structure [BH] is added to the S_list (Figure 4j).

In place I, structure [HI] is added to the S_list (Figure 4k) and task $\delta 1$ is erased. The type 1 and type 5 mapping rules that record the pairs (B, $\delta 1$) and (H, $\delta 1$), and the structures [B, H]/$\delta 1$ and [H, I]/$\delta 1$ are created. As the navigation between B and H involves several detections of landmarks and subtasks, two type 3 mapping rules and one type 4 are also created, that record the pairs (B, $\delta 2$), (D, $\delta 3$) and (F, H).

From place I, the animat resumes pursuing its initial goal. However, in a purely exploratory mode - i.e., in the absence of any such goal - it might be motivated to explore further the obstacle it is skirting around and, thus, to turn to its left. This would consequently create the task $\delta 4$ shown on Figure 3 and trigger the creation of a type 6 rule, which would record the pair (I, $\delta 4$).

Whatever the case, after the map has been learned, it appears that landmark D, for instance, can be detected as a marker landmark at the beginning of a structure [DE] within context $\delta 3$, or as a task-erasing landmark ending structure [CD] within context $\delta 2$, or as an unsatisfying landmark belonging to structure [BH] within context $\delta 1$. It will be shown in the next paragraph that such different ways of recognizing landmark D substantiate each other and contribute to make the animat's self-positioning easier.

4 Self-positioning

Having learned a map of the environment, the animat must be able to use it to position itself, even if its position and orientation estimates are noisy. In this paragraph, a simplified version of the MonaLysa's self-positioning procedure will be described, assuming that, although no errors were made during a preliminary map-building stage, the animat must now position itself under conditions of noisy position estimates. In the following paragraph, experimental results obtained under more realistic conditions will be presented.

In MonaLysa, the self-positioning procedure relies upon the management of a H_list within the spatial information processor. Such a list is associated with each skirting task triggered and records all the pairs (Ai, δi) on the map that match the animat's *current hypothesis* about its position within the environment.

The procedure also entails the management of a PH_list of *position hypotheses* within the place-recognition module, each such hypothesis being characterized by an *error estimate* and a *confidence estimate*. At every time step, one of these hypotheses is the animat's current hypothesis and designates the place where the animat estimates it is the most likely to be. Nevertheless, the animat also considers that it could be situated in other places, although the corresponding hypotheses are less likely to be true. New hypotheses are posted each time a marker is detected in the context of a skirting task and matches a given landmark on the map. This event occurs when a first marker is detected and triggers type 1, 2 or 6 mapping rules, or when a structure is detected and triggers type 5 mapping rules. When the distance between such an hypothesized position and a place already recorded on the map - that is, a place recorded by mapping rules of types 1, 2, 5, or 6 - doesn't exceed the animat's length, the memorized place replaces the hypothesized position, and the error estimate associated with this position is reset to the memorized error, i.e., to 0 if the map has been learned without error. Between two such reset episodes, the error estimates of each position hypothesis increase with the distance actually traveled by the animat in a manner that will not be described here for lack of space.

As to the confidence estimate associated with each position hypothesis, it increases when the corresponding position is replaced by a landmark already recorded on the map, the importance of this effect being greater when the landmark belongs to a structure than when it doesn't. Moreover, supports provided by individual components of a given structure can modulate this effect, when mapping rules of types 3 and 4 are triggered. The confidence estimate associated with each position hypothesis decreases when no equivalence with a memorized landmark can be found. However, the exact way such confidence estimates are reset will not be described

herein.

Figures 5 and 6 help to describe the self-positioning procedure from a specific example. They assume that the animat, after a preliminary exploratory stage, has detected 4 obstacles in its environment and built the corresponding map, according to the mapping procedure previously described. Such a map involves landmarks B',...N', B",... N" and skirting lines like $\delta 1$', ...$\delta 4$' ,$\delta 1$", ...$\delta 4$" (Figure 5). It also assumes that the animat, currently in a positioning phase, arrives at place A', but erroneously assumes it has reached place A. Thus place A is its current positioning hypothesis - which is supposedly characterized by an error of 250 and a confidence level of 100 - and the top of the context-manager pile contains task $\delta 1$ (Figure 6a).

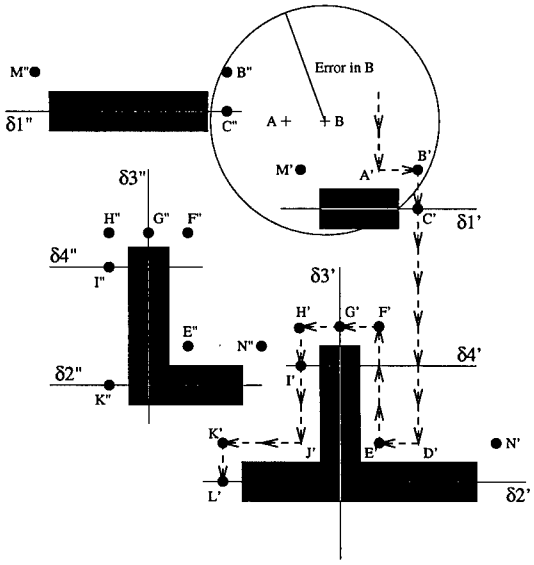


Figure 5: The use of the cognitive map for self-positioning. Filled circles refer to landmarks recorded in the map

In A', the animat turns left and reaches place B', which is characterized as a satisfying landmark B. B is thus added to the LC_list associated with $\delta 1$. Because the coordinates of the animat in places B' and B" fall within the error margin of place B, because the animat's orientation in these places is the same, and because all these places are characterized as landmarks of the same type, places B' and B" are said to *match* place B. Likewise, because the origins of the vectors defining lines $\delta 1$' and $\delta 1$" fall within the error margin associated with the origin of the vector defining $\delta 1$, and because the corresponding orientations are the same, tasks $\delta 1$' and $\delta 1$" match task $\delta 1$. On the contrary, place B doesn't match place M' because, although they are both satisfying landmarks, the animat's orientation in these places differ. Neither does place B match place C", because the latter is a task-erasing landmark and because the animat's orientations in both places differ. Thus, 2 mapping rules of type 1 that have their condition parts matching the current situation are triggered and post pairs (B', $\delta 1$') and (B", $\delta 1$") on the H_list of $\delta 1$. The current positioning hypothesis becomes B, which is characterized by an error of 251, and two new positioning hypotheses, B' and B", are added to the PH_list and are both characterized by an error of 0 and an initial confidence level of 25. The confidence level of B, which is not recognized on the map, decreases to 90 (Figure 6b). In other words, as the animat moves away from a landmark recorded in its map, its position error increases and its confidence in the current positioning hypothesis decreases.

From B' the animat moves to C', where it updates its 3 position hypotheses, and records that, although it has probably moved from B to C [1], it might have moved from B' to C*' or from B" to C*" as well.

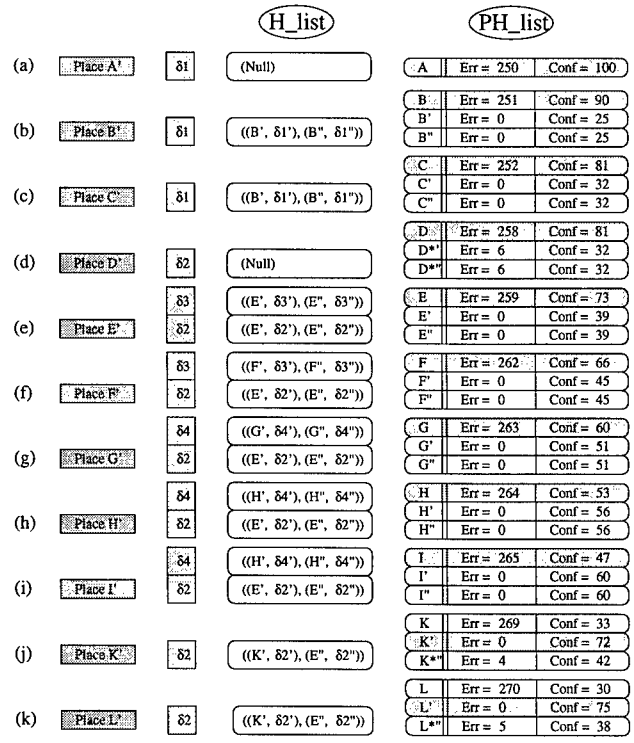


Figure 6: The management of the H_list and PH_list during self-positioning. Shaded components of the PH_list correspond to the animat's current position hypotheses.

At C', the current position C is detected as a task-erasing landmark, and structure [BC] is posted on the S_list of $\delta 1$. Current conditions in C match those of C' and C", allowing the triggering of 2 mapping rules of type 5, which suggest that the place the animat is currently in belongs to structures [B'C'] or [B"C"]. Because place B matches places B' and B", because place C matches

[1] Place C, like several other places that are mentioned in the text, is not shown on Figure 5.

places C' and C", and because the relative displacement the animat made between B and C matches the relative displacement it made from B' to C' and from B" to C", structures [B'C'] and [B"C"] match structure [BC], thus supporting the hypothesis that place C could, in fact, be either place C' or C". These hypotheses are accordingly posted to the PH_list. Because positions of C*' and C"* are respectively no more distant from places C' and C" than the animat's length, the positions of C'* and C*" can be replaced by those of C' and C", and their error estimates can be reset to zero. The confidence estimates of C' and C" thus increase to 32 when the confidence estimate of B decreases to 81 (Figure 6c).

Then the animat travels from C' to D', updating its 3 position hypotheses, and records that, although it has probably moved from C to D, it might have moved from C' to D*' or from C" to D*" as well (Figure 6d). Then it turns to the right and generates skirting task $\delta 2$. At place E', the place E is characterized as an unsatisfying landmark that matches places E' and E" and task $\delta 3$ is created. Type 4 mapping rules post pairs (E', $\delta 2$'), (E", $\delta 2$"), (E', $\delta 3$') and (E", $\delta 3$") respectively on the H_list of $\delta 2$ and $\delta 3$, and the two place hypotheses E' and E" are posted on the PH_list. These hypotheses replace the current hypotheses E*' and E*", and their confidence estimates increases to 39, while the confidence estimate of E decreases to 73 (Figure 6e).

At place F', task-erasing landmark F is matched with F' and F", and structure [EF] is matched with structures [E'F'] and [E"F"]. Thus, the confidence estimates of the hypotheses associated with F' and F" continue to increase, while the confidence estimate of the hypothesis associated with F continues to decrease (Figure 6f).

From F' to G', H' and I', the PH_list maintained by MonaLysa is updated according to Figures 6g, 6h and 6i.

When the animat arrives at K', it supposes itself to be at K, and structure [EK] = ((E, $\delta 3$), (G, $\delta 4$), (I, K)) is posted on its S_list. As such a structure matches [E'K'] but not [E"K"], it turns out that K*' can be replaced by K' and that its error estimate can be reset to 0. Because the confidence in hypothesis K' exceeds a given threshold level (i.e., 50) and because the confidence associated with K and K*" does not, the animat's current hypothesis switches from K to K' (Figure 6j). In other words, the animat has positioned itself accurately in its environment.

Had structure [E"K"] matched the detected structure [EK] in the presence of a high noise level, then the supports posted during navigation from E' to K' by the three components of the detected structure [EK] would nevertheless have been more likely to contribute to a greater increase in the confidence estimate of K' than to that of K". Indeed, components (E, $\delta 3$) and (G, $\delta 4$) of [EK] would match the condition parts of type 3 rules that have been created within the contexts of the exploration of [E'K'] and [E"K"] and would receive support from both categories of rules. On the contrary, component (I, K) would be more likely to match the condition part of a type 4 rule created within the context of the exploration of [E'K'] than within the context of [E"K"].

It should be understood that self-positioning success relies upon the fact that MonaLysa has managed simultaneously several hypotheses about the animat's moves and that some of these hypotheses got support every time a given landmark or structure in the environment was matched with a landmark or structure in the map. Thus, the hypothesis that the animat traveled from E' to K' got support not only from the matching of E with E', of F with F', of G with G', of H with H', of I with I', and of K with K', but also from the matching of [EF] with [E'F'], of [FG] with [F'G'], of [GH] with [G'H'], of [HI] with [H'I'], and finally from the matching of the whole structure [EK] with [E'K']. Had the animat failed to recognize a given landmark on the E'K' trajectory, like landmark I' for instance, the supports from the matchings of I with I', of [HI] with [H'I'] and of (I, K) in [EK] with (I', K') in [E'K'] would have been lost. Nevertheless, the other supports would have been retained, thereby preventing the hypothesis that the animat actually moved from E' to K' from being abandoned. Thus the management of hierarchical structures affords the self-positioning procedure of MonaLysa with interesting robustness features.

5 Experimental results

To demonstrate the efficiency of the mapping and positioning procedures just described in a more realistic context, both procedures have been managed in parallel. The animat has been placed in an initial random position within the environment of Figure 7 and left free to explore it. It has not been provided with any overall goal, but its satisfaction was assumed to be greater when it was moving straight on, as opposed to turning right or left. Its orientation and position estimates were assumed to be correct and the distance traveled after each move was assumed to be equal to the animat's length. Furthermore, a specific procedure, not described herein, allowed it to learn to differentiate between obstacles around which it could skirt, and obstacles that merely delineated the environment's boundaries. Figure 7 also shows parts of the cognitive map thus elaborated after 50000 elementary moves. In particular, although this map doesn't represent any of the skirting lines and structures that have been detected in the environment, it shows the various landmarks and their topological links that the animat has memorized.

Figure 8 shows parts of the cognitive map elaborated when the animat's position estimates were assumed to be noisy and to increase with the distance traveled. Thus, after each elementary move beyond a given place, the error on the corresponding position estimate was sup-

posed to be a random normal variate, with a mean of 0, and a standard deviation proportional to the animat's size. After n such elementary moves, the corresponding standard error was that of a sum of normal variates, i.e., it was proportional to $\sqrt{n}*animat's_length$. Because such errors were taken into account in both the mapping and positioning processes, the matching of two places was considered successful when the corresponding error margins intersected. Under such conditions, the animat wandered in its environment and, after each move, updated its cognitive map and its position estimate. Figure 8 shows parts of the cognitive map elaborated after 50000 elementary moves.

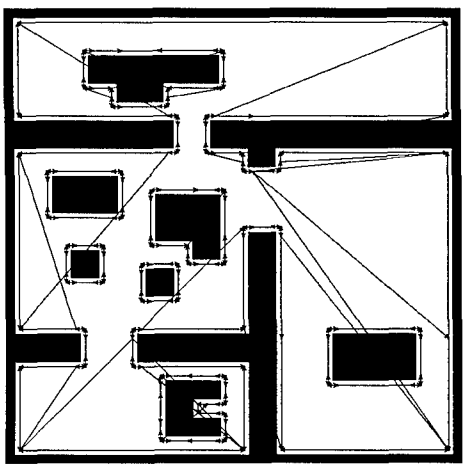


Figure 7: A square environment explored by an animat endowed with the MonaLysa control architecture. The size of the animat is 15 pixels, the size of each side of the environment is 735 pixels. Parts of the cognitive map elaborated by MonaLysa in the absence of noise are shown on the picture, as vectors coding topological links between pairs of landmarks

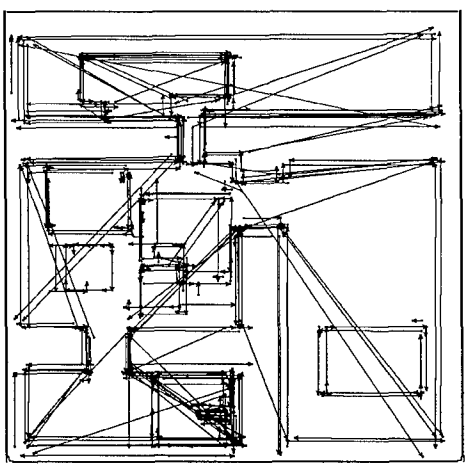


Figure 8: The cognitive map built when the standard deviation of the noise on the position estimate after each elementary move was assumed to be equal to 10% of the animat's length.

Figure 9 shows how the distance between the animat's current position estimate and its actual position varies over time in noisy conditions, when the mapping procedure is used and when it is not. It thus appears that its cognitive map allows the animat to control its positioning error. In particular, this error seldom exceeds twice the animat's length and, when such an event does occur, the error decreases to its basic level in a few moves.

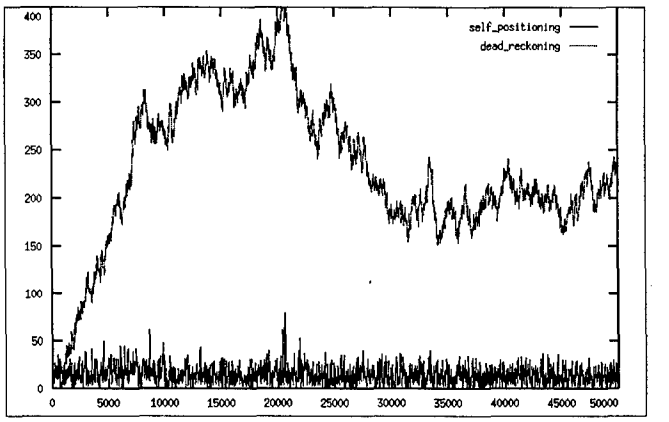


Figure 9: Evolution of the distance between the animat's current position estimate and its actual position, during the construction of the cognitive map of Figure 8. The upper curve correspond to results obtained without using the map, the lower one corresponds to results obtained when the map was used. Distances in pixels (ordinate) versus successive elementary moves (abscissa)

Additional experiments indicate that the control of the positioning error is robust at reasonable noise levels. However, and as might be expected a priori, map learning and self-positioning become impossible beyond a certain noise threshold, roughly equal to 20 % of the animat's length.

6 Discussion

It has been demonstrated here and elsewhere that the MonaLysa control architecture enables an animat equipped with a rudimentary sensory-motor apparatus to explore its environment, to extricate itself from dead-ends with arbitrary complicated shapes, to build a cognitive map of its environment, to accurately estimate its current position, to plan trajectories that avoid obstacles and that lead to a given externally-specified or autonomously-generated goal. Such capabilities qualify MonaLysa as a *cognitive* architecture and the animat as a *motivationally autonomous system* (McFarland and Bösser, 1993; Donnart and Meyer, 1996). They rely upon the use of production rules that take into account, not only the animat's current sensory information, but also their " internal context ", which codes the animat's additional knowledge about the current situation. In the

case of the reactive rules, this knowledge concerns the direction of the current goal. In the case of the planning rules, it concerns the nature of the current task the animat tries to accomplish. In the case of the mapping rules, it concerns the various tasks associated with markers and structures the animat has detected in its environment. Moreover, the fact that this internal context has a hierarchical structure confers several advantages on MonaLysa. Besides expediting the learning of the planning rules as described in (Donnart and Meyer, 1996), it has been shown here that it facilitates the self-positioning procedure and makes it relatively robust with respect to noise.

It has also been shown here that such robustness makes it possible for MonaLysa to detect, memorize and recognize landmarks in the environment, although the corresponding procedures do not rely on the information provided by the animat's proximate sensors. Actually this information is only used by the reactive rules of MonaLysa for the purpose of skirting around obstacles and of escaping from dead-ends, and place recognition depends only upon the proprioceptive position, orientation, and satisfaction estimates. Such a characteristic is in sharp contrast with many other realizations that implement place-recognition capabilities. For instance, several such realizations draw upon biology (Zipser, 1986; Cartwright and Collett, 1987; Muller et al., 1991; Burgess et al., 1994; Bachelder and Waxman, 1994) and implement a neural network in which the firing of some sensory neurons, tuned to the features of some landmarks sensed in a given place, triggers the firing of a specific *place cell* that codes for this place. Likewise, in (Kuipers and Byun, 1991), distinctive places are defined as the local maxima of specific functions that are defined over the various sensor readings of the animat, and, in (Mataric, 1992), landmarks are characterized as features in the world that have physical extensions reliably detectable over time. Few other realizations resort to proprioceptive information for place recognition. A specific example is provided in (Nehmzow and Smithers, 1991), where landmarks are characterized as convex or concave corners, such that, if the time a robot needs to turn towards a wall exceeds a certain threshold time, a convex corner is detected. Conversely, if the time it takes the robot to get away from a detected obstacle exceeds a certain threshold time, a concave corner is detected.

The route-following navigation strategy of MonaLysa could also be contrasted with other realizations which do not fully exploit the information encoded in their cognitive maps. In such realizations, indeed, a specific place is usually recognized because it has been reached by a specific move from another specific place, but no account is taken of the hierarchical structures that MonaLysa manages. As an example, in (Nehmzow and Smithers, 1991), a specific concave corner can be recognized because it has been reached from a convex corner a given distance away, but not because this specific concave corner belonged to a characteristic structure that linked, for instance, three succeeding convex corners to the concave corner in question. It is certainly because such realizations do not take advantage of the robustness that hierachical contexts afford to mapping and positioning procedures that their exploration capacities are much more limited than those of MonaLysa. The Nehmzow and Smithers's robot (Nehmzow and Smithers, 1991) cannot position itself in the environment if it doesn't keep following its boundary walls. Likewise, Mataric's robot (Mataric, 1992) and Kuipers and Byun's animat (Kuipers and Byun, 1991) rely heavily upon their wall-following and corridor-following exploration strategies to avoid getting lost. In contrast, the animat described in this paper is able to navigate randomly from one obstacle to another and to position itself. However, the present work and that of Mataric have in common the fact that place-recognition not only depends upon topological information, but also upon metric information about distances and orientations. Likewise, the present work and that of Kuipers and Byun have in common the fact that place-recognition depends upon topological links that are indexed by the animat's control strategy. With MonaLysa, an actual move between two places can be matched to a possible move on the map provided they occur within the same skirting-task context. In Kuipers and Byun's approach, two moves can be matched if they are actuated by the same *follow-the-midline* or *move-along-object-on-right* control strategies.

Although MonaLysa seems able to cope with less carefully designed environments than those that have been used in other realizations, it is nevertheless true that its current mapping and self-positioning capabilities would be lost if the environment contained non-polygonal obstacles. To allow the management of arbitrarily shaped obstacles, future research will be directed towards the use of more general skirting tasks than mere line crossing.

Future research will also be directed towards the possibility of recognizing whole obstacles and not only single landmarks or structures. This faculty will introduce an additional hierarchical level in the mapping and self-positioning procedures, and thus enhance the animat's cognitive capacities. In the present implementation, although the animat uses relative distances and orientations to self-position itself along the external contours of a given obstacle, it resorts to absolute distances and orientations to position itself when it first encounters a new obstacle. The possibility of characterizing whole objects in the environment will make it possible to evaluate all positions and orientations relatively to each object in the environment, and thus to get rid of any reference to the animat's initial position.

The planning capacities of MonaLysa have already

been demonstrated with a real robot (Donnart and Meyer, 1996). The present work has used simulations to demonstrate its mapping capacities. Future research will aim at combining the planning and mapping capacities of the architecture, first within a simulation framework, then with a robotic application. It will also aim at translating the actual rule-based implementation of MonaLysa into a biologically more realistic neural network architecture.

7 Conclusions

It has been shown here that the MonaLysa control architecture endows an animat with map-building and self-positioning capabilities that are robust with respect to noise. Such capabilities rely upon mapping and positioning procedures that take into account specific landmarks and structures in the environment. This approach is original and exhibits several advantages in comparison with other works that have similar objectives. In particular it can be used in environments that need not be as carefully designed as usual. In the future, it will be extended so that, beyond single landmarks and structures, it can characterize and recognize whole objects. Future work will also target the management of general skirting tasks that will hopefully cope with arbitrarily shaped obstacles. It will also be extended to the implementation of MonaLysa on a Khepera robot and to the demonstration that its mapping and planning capacities do actually work in the real world.

References

Bachelder, I.A. and Waxman, A.M. (1994) "A neural system for qualitative mapping and navigation in visual environments." *In Proceedings of the PerAc Conference: from Perception to Action. IEEE Soc. Press.*

Burgess, N., Recce, M. and O'Keefe, J. (1994) "A model of hippocampal function." *Neural Networks 7(6/7) :1065-1081.*

Cartwright, B.A. and Collett, T.S. (1987) "Landmark learning in bees. Experiments and models." *Journal of Comparative Physiology A 151 :521-543.*

Chatila, R. and Laumond, J. (1985) "Position referencing and consistent world modeling for mobile robots." *In IEEE Proceedings of Int. Conf. on Robotics and Automation. pp. 138-145.*

Donnart, J.Y. and Meyer, J.A. (1994) "A Hierarchical Classifier System Implementing a Motivationally Autonomous Animat." *In Proceedings of the 3rd Int. Conf. on Simulation of Adaptive Behavior. The MIT Press/Bradford Books, pp. 144-153.*

Donnart, J.Y. and Meyer, J.A. (1996) "Learning Reactive and Planning Rules in a Motivationally Autonomous Animat." *IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics, Vol. 26, No. 3.*

Gallistel, C.R. (1990) "The Organization of Learning." *The MIT Press.*

Kuipers, B.J. (1982) "The ' map in the head ' metaphor." *Environment and Behavior 14(2) :202-220.*

Kuipers, B.J. and Byun, Y.T. (1991) "A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations." *Robotics and Autonomous Systems 8 :47-63.*

Latombe, J.C. (1991) "Robot motion planning." *Kluwer Academic Publishers.*

Mataric, M.J. (1992) "Integration of Representation Into Goal-Driven Behavior-Based Robots." *IEEE Transactions on Robotics and Automation 8(3) :304-312.*

McFarland, D. and Bösser, T. (1993) "Intelligent Behavior in Animals and Robots." *The MIT Press/Bradford Books.*

Muller, R.U., Kubie, J.L. and Saypoff, R. (1991) "The hippocampus as a cognitive graph." *Hippocampus 1(3) :243-246.*

Nehmzow, U. and Smithers, T (1991) "Mapbuilding using Self-Organising Networks in 'Really Useful Robots'." *In Proceedings of the 1st Int. Conf. on Simulation of Adaptive Behavior. The MIT Press/Bradford Books, pp. 152-159.*

Nehmzow, U. (1995) "Animal and robot navigation." *Robotics and Autonomous Systems 15 :71-81.*

O'Keefe, J. A. and Nadel, L. (1978) "The Hippocampus as a Cognitive Map." *Oxford University Press*

Penna, M.A. and Wu, J. (1993) "Models for map building and navigation." *IEEE Transactions on Systems, Man, and Cybernetics 23(5) :1276-1301.*

Poucet, B. (1993) "Spatial cognitive maps in animals : New hypotheses on their structure and neural mechanisms." *Psychological Review 100 :163-182.*

Riolo, R. L. (1991) "Lookahead planning and latent learning in a classifier system." *In Proceedings of the 1st Int. Conf. on Simulation of Adaptive Behavior. The MIT Press/Bradford Books, pp. 316-326.*

Roitblat, H. L. (1994) "Mechanism and process in animal behavior: Models of animals, animals as models." *In Proceedings of the 3rd Int. Conf. on Simulation of Adaptive Behavior. The MIT Press/Bradford Books, pp. 144-153.*

Schmajuk, N.A. and Thieme, A.D. (1992) "Purposive behavior and cognitive mapping : A neural network model." *Biological Cybernetics 67 :165-174.*

Scholkopf, B. and Mallot, H.A. (1995) "View-Based Cognitive Mapping and Path Planning." *Adaptive Behavior 3(3) :311-348.*

Thinus-Blanc, C. (1988) "Animal spatial cognition." *In Weiskrantz, L. (Ed.). Thought without language. Clarendon.*

Tolman, E.C. (1948) "Cognitive maps in rats and men." *The Psychological Review 55 :189-208.*

Zipser, D. (1986) "Biologically plausible models of place recognition and goal location." *In Rumelhart, D.E. and McClelland, J.L. (Eds.). Parallel Distributed Processing. The MIT Press/Bradford Books.*

Adaptive animat navigation based on a flexibility model for the environment

Peter Veelaert and Herbert Peremans
University of Ghent, Dept. ELIS
Parallel and Intelligent Systems
Sint-Pietersnieuwstraat 41, B-9000 Ghent
veelaert@elis.rug.ac.be, peremans@elis.rug.ac.be

Abstract

In this paper we propose a unified framework for local animat navigation and position localization. This framework is based on the use of flexibility maps of the environment. Flexibility maps contain both the current knowledge of the animat about the positions of the objects in its environment as well as the information required to calculate its future path. The major benefit of using flexibility maps is that several navigation tasks can be incorporated elegantly into a single framework. In addition, we found that the simple control laws derived from this approach generate paths that look very natural in realistic circumstances, adding to the biological plausibility of the approach. We will show that a flexibility model increases the adaptivity of an animat to changes in its environment by allowing the use of families of paths instead of single paths, and by allowing the derivation of entirely new paths from the animat's memory of previously learned paths.

1 Introduction

In this paper we propose a unified framework for modeling an animat and its environment which allows the systematic design of navigation behaviors. This framework, which we call a flexibility model, keeps track of all the possible motions of both the animat itself as well as of all the other objects in its environment. The use of a flexibility model has been a direct result of our investigations into the most basic task for a navigating animat: the problem of localization. That is, the animat must have some notion of its own position relative to a set of landmarks. This problem may take several forms. It may either concern the localization of a unique relative position, or more generally, the localization of a path, or a class of paths along which the animat is moving. For example, the animat may have to know that it is entering a room although knowledge about its precise current position, or even its precise course may not be important. In all these cases, if formulated accurately, a self-localization task invariably leads to a flexibility model. We conjecture that a flexibility model is a natural way of representing all the information needed by an animat to successfully navigate through its environment. The plausibility of this claim in the case of landmark based animal, including human, navigation is based on a number of interesting properties of the paths defined by the flexibility model.

(i) The paths generated while performing a number of realistic navigating behaviors such as "entering a door", "following a wall", "turning left at the nex corner", etc. look very natural as can be seen from Fig. 1. As such the flexibility model could be an alternative for potential fields to implement the navigation schemata described by Arkin [1].

(ii) The paths can be described using relative positions only, there is no need for absolute positions. Hence, the flexibility model fulfills one of the essential conditions of qualitative navigation approaches in mobile robotics [6, 7].

(iii) Following such a path can be accomplished by a simple control law requiring only easily obtainable sensor information. Even animals with limited cognitive powers should be able to implement this scheme.

(iv) The paths can be made to pass exactly through a number of checkpoints equal to the number of landmarks present. Hence, landmark based homing or more general landmark based goal seeking as performed by many animals, bees, rats, pigeons [4, 3] to name but a few, is easily accomplished using the flexibility model.

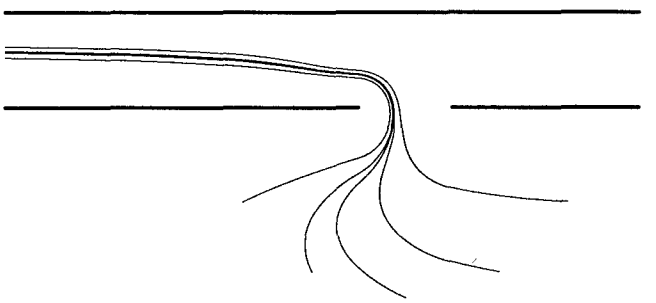


Figure 1: The paths generated by the flexibility model when entering a corridor through a door and turning to the left, the three walls are used as landmarks.

(v) It will be shown that adapting these paths to changing circumstances, the addition or removal of landmarks for example, can be done incrementally. Imagine for example a flying animat that is currently using two landmarks for its navigation along part of its course. Suppose that the next time the animat flies along this course we add a third landmark. Obviously, the animat can use this additional landmark to upgrade the complexity and the accuracy of its course. According to the flexibility model however, this upgrade will be the result of a smooth adaptation of the old path.

These five properties will be discussed in the remainder of this paper. Note that, although the concepts used in this paper can be given a sound mathematical basis ([2, 5, 9]), we shall limit the use of mathematics as much as possible. In fact, most of the ideas will seem to be rather natural and intuitive.

2 Flexibility models as representations for the environment

We want to show that there exists a deep and interesting relation between the constrained motion of parts of mechanical constructions and the navigation of animats. First we will clarify this link and subsequently, in the remainder of the paper, it will be used to solve some typical navigation problems. In this section we will restrict ourselves to describing the flexibility model regardless of how it was acquired by the animat. In a later section we will explain how such a flexibility model can be built up, and how it can be adapted.

Let us assume that we have an animat moving around in an environment, and that this animat has sensors so that it can measure a relationship between its own position and an object in the environment. Usually, these relationships consist of the distance between the animat and an object or the bearing of the line of sight from the animat to the object. In this navigation context many naturally arising questions can be answered by transforming them first into questions about a mechanical analogue.

Position localization. We start with a simple example. Suppose that the animat has measured three distances to three different objects in its neighborhood. Can it determine its position relative to those three objects uniquely? If each measured distance (=relationship) is represented in the mechanical analogue by a rod with a fixed length and the objects in the environment, including the animat, are represented by joints this situation corresponds with a mechanical construction looking like Figure 2(a). The distances that have been measured are shown with heavy lines. Since this mechanical framework fixes the different movement capabilities of its constituent parts and since these parts correspond directly to elements from the animat's environment we refer to this mechanical framework as *the flexibility model of the environment.* Note that this model contains the knowledge the animat has about its environment after measuring three distances, and nothing more than that, i.e. the animat has not performed any labeling or recognition of the landmarks.

It is obvious that in the above example parts of the construction are still able to move with respect to each other, i.e. the construction is not rigid. In fact, Figure 2(b) shows a second configuration of the same construction which corresponds equally well with the animat's current knowledge. Hence, from the lack of rigidity of the mechanical construction we can conclude that to determine its relative position uniquely, the animat must find out more about its environment. Furthermore, the mechanical construction shows that it is sufficient to measure, for example, the distance between the first and the second landmark, and the distance between the first and the third landmark, as shown in Figure 2(c). Since in that case the construction would become rigid, and, returning to our original localization problem, all relative distances would become fixed. In fact any two measurements of distances that are independent of the previous measurements would suffice.

Note that once the construction is rigid the animat can lookup the environment configuration in its memory to check whether this is a known configuration. If

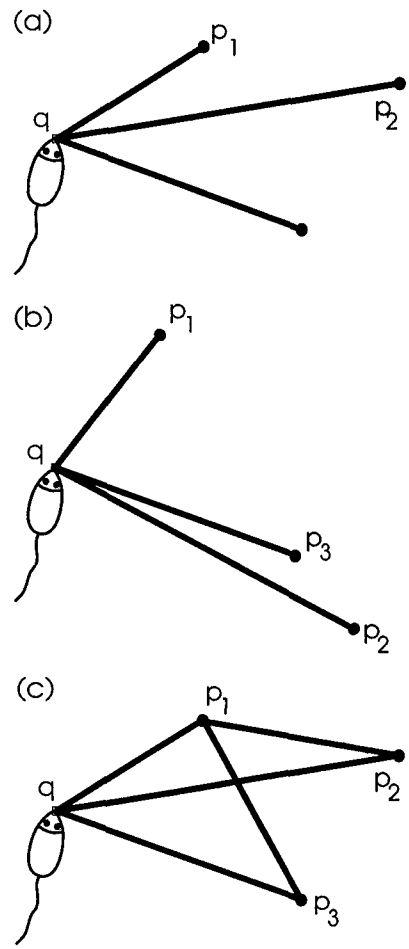


Figure 2: The position localization problem is related to rigidity.

so, it can go from its current position to any memorized position defined with respect to this configuration. Such a scheme could be used by animals relying on landmarks to reach a particular goal location, i.e. finding a feeding place, returning home. As such, the localization scheme described here would be a generalization of the scheme described by Zipser in [8] to arbitrary numbers of landmarks. In that paper a localization scheme is described that could explain how rats are able to locate an underwater platform in a tank filled with milky water, making the platform itself invisible, by using three external landmarks.

Clearly, other position localization problems, making use of sensors that measure angles instead of distances for example, would be variations on the same theme. *Hence, we conclude that to localize itself an animat has to measure geometric relationships until it has removed enough degrees of freedom from its flexibility model such that no motion, except for a trivial motion of the entire structure, is possible anymore.* With trivial motions we denote rotations and translations that cause identical position changes for all parts of the structure, i. e. such motions do not change the relative positions of the different parts of the structure.

Based on this result, many interesting conclusions can be drawn from reasoning about the rigidity of flexibility models corresponding with specific localization strategies. For instance, suppose the animat measures the distances between its own position and the landmarks at different points along its path through the environment, as shown in Figure 3(a). One can readily see that the corresponding mechanical framework will never become rigid, no matter how many distances are measured by the animat. Hence, it follows that the animat must measure either the distance between two of the landmarks or the distance between two of its own positions, as shown in Figure 3(b). This result is readily generalized to situations where there are more than two landmarks. This yields an interesting conclusion. *To localize itself, an animat must either be able to measure distance and bearing of the landmarks in its environment, or it must be able to measure the distance that it has been traveling along a straight segment of its path.*

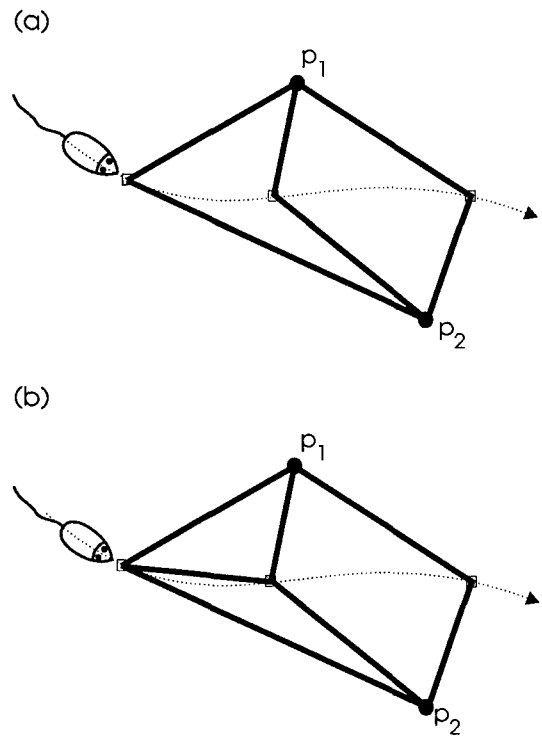


Figure 3: Distances that the animat must measure for position localization.

Path localization As formulated above, the position localization problem can be solved by performing a sufficient number of independent measurements until the mechanical analogue becomes rigid. Inspired by this analogue we now propose another principle: *In many cases the motion of an animat is the simplest possible extension of position localization.* That is, we extend the principle of position localization as we have formulated it using flexibility models, so that it will now also incorporate the animat's motion. The advantage of doing so is that, as far as the animat is concerned, motion and position localization become basically the same operation: attaching itself to its environment through measuring certain geometric relations. This attachment results in a rigid mechanical analogue in the case of position localization and in a flexible, allowing one degree of freedom, mechanical analogue in the case of path localization. This remaining degree of freedom makes it possible for the animat to traverse its path.

Suppose that an animat wants to move from point A to point B. At first sight, this can be done by first specifying a path and then solving the complete position localization problem for every point along this path. In most cases, however, this will not be necessary. It is usually sufficient for the animat to know that it is still on the right path and heading in the right direction, regardless of its exact position. Returning to the mechanical analogue, we will no longer remove all the degrees of freedom, which is the primary objective for position localization, but instead we will only remove some of them and use the remaining degrees of freedom to specify the motion of the animat. In that case, the animat, while moving around, may not know its exact position, but it will know exactly on which path it is. In other words, position localization will be replaced by path localization, which is less demanding.

Figure 4(a) shows a simple example of path localization. Suppose that the animat has determined its initial position q uniquely with respect to the three fixed landmarks p_1, p_2, p_3. Suppose now that the animat measures the distance to p_1, and that it starts moving around while keeping this distance fixed. Although the animat may then no longer know its current position, it does know that it is moving on a circular path around p_1. In terms of our mechanical analogue, we have a rigid construction of three landmarks, and an animat attached with a single rod to the first landmark.

This kind of animat motion is still very restricted. How can we introduce more complex paths? It turns out that to allow the specification of more complex paths we have to introduce into the mechanical construction, in addition to the rods, combinations of strings with a fixed total length. Figure 4(b) shows an example where the animat is attached with three strings to the three landmarks (represented by thin lines). It now tries to keep the sum

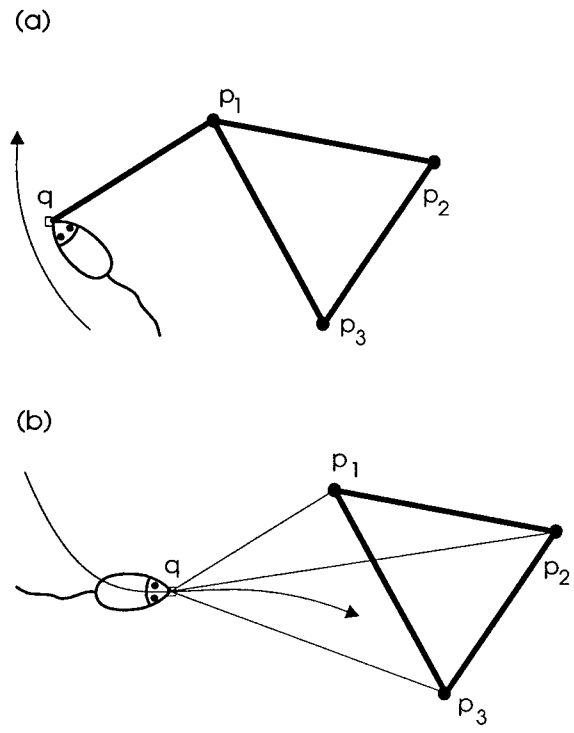


Figure 4: The mechanical analogue for path following.

$$\alpha_1|p_1 - q| + \alpha_2|p_2 - q| + \alpha_3|p_3 - q| \qquad (1)$$

fixed, where the α_i are real numbers, and where $|p_i - q|$ is the distance between p_i and q. Figure 4(b) shows a possible motion. Henceforth, all the motions described in the remainder of this paper will be based upon structures such as the one shown in Figure 4(b). Note that when the α_i are positive rational numbers the combination of strings with a fixed total length can be straightforwardly implemented by a pulley.

3 Path following based on the flexible framework

As described in the previous section the flexibility model contains all the information the animat possesses about its environment including a description, as summarized by Eq. (1), of its own movement through this environment. We now show how it is possible for an animat to stay on its path as specified by the flexibility model.

If, according to its current world view, the animat tracks a single landmark, Fig. 4(a), motion control is easy. It suffices to find the line of sight of the landmark, and to move perpendicular to it. The animat will circle around the landmark. It turns out that the more general case, with an animat observing several landmarks, is only slightly more difficult; again, it will be sufficient to determine the lines of sight of the landmarks. Let r_i denote the distance between the animat and the i-th landmark. Since the definition of a motion, i.e. Eq. (1), involves relative distances to fixed landmarks only, the animat can move along its path by a simple control algorithm. At every point along the animat's path we must have $\sum \alpha_i r_i = c$, where c is some fixed number. To follow the path it is sufficient to keep this relation valid while the animat is moving, although the relative distances r_i themselves will be changing. Hence, taking derivatives with respect to time, we must have that

$$\sum \alpha_i \dot{r}_i = 0, \tag{2}$$

where $\dot{r}_i$ denotes the velocity of the ith landmark relative to the animat and along its line of sight. To calculate the traveling direction we choose an arbitrary orthogonal xy-coordinate frame, for example, the frame shown in Figure 5. Let $q = (x, y)$ denote the momentary position of the animat. Then the relative velocity $\dot{r}_i$ of a landmark can be written as

$$-\dot{r}_i = \dot{x}\cos\beta_i + \dot{y}\sin\beta_i,$$

where β_i is the angle between the x-axis and the line of sight of the landmark as seen by the animat. Substituting this expression into (2), it follows that

$$\frac{\dot{x}}{\dot{y}} = -\frac{\sum \alpha_i \sin\beta_i}{\sum \alpha_i \cos\beta_i}. \tag{3}$$

According to this result, to stay on the current path it is not necessary to know the distances r_i, knowing the bearings of the different lines of sight suffices. For visual sensors this is a useful property, since for such a sensor, especially the more primitive compound eyes of insects [11], it is easier to obtain angular information than range information. The above expression can be used in a simple control loop that links the angle information about the landmarks to a motor control vector. In fact, in a very simple control structure, we can link the neurons of a silicon retina directly to a set of motor control neurons.

4 Adaptive behavior based on the flexibility model

We have seen that the flexible framework view of the world leads to simple motion control laws. In fact, according to the flexibility model, animat motion is completely integrated into the representation that the animat has of its environment. In this section we discuss how this flexibility model of the world can be adapted in a changing environment, and therefore how the changing environment will influence the motion of the animat. The basis for adaptation will be the capability of local path planning. In particular, we will see that an animat can calculate the weights of the strings once it has chosen a set of local checkpoints along the path. Next, it can combine simple paths that involve only a few checkpoints to obtain more complicated paths. Thus on the one hand there is the possibility for the animat to build up its course smoothly while gradually taking into account more and more landmarks in an effort to optimize its course. On the other hand, there is the possibility of smooth degradation, when one or more landmarks cannot be found anymore, and the animat has to fall back on the simpler control laws again.

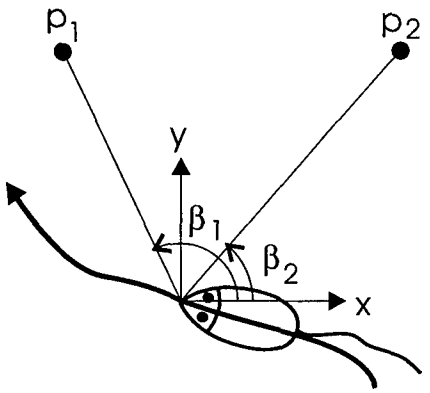


Figure 5: Animat following a path by determining the lines of sight of the landmarks.

In this process of smooth building up and degradation one particular family of paths turns out to be very useful. We will show that we can decompose local path planning into path planning that involves entering or passing gates only, i.e. paths that use only two landmarks.

4.1 Local path planning by choosing checkpoints.

We first discuss how local path planning can be accomplished by choosing some checkpoints along the path. More specifically, given an environment with an animat and a set of landmarks, we will choose a number of checkpoints in this environment, and then we will test whether there is a flexibility model for this environment

with strings attached to the animat such that the motion defined by the flexibility model passes through the checkpoints that we have chosen.

We first discuss the case where the number of checkpoints is equal to the number of landmarks. Suppose that we have an environment with an animat q, and three landmarks p_1, p_2, p_3, as shown in Figure 6. We choose three different checkpoints q^1, q^2, q^3. These checkpoints correspond to three different positions q^1, q^2, q^3 of the animat in its environment. If the animat is at initial position q^1, can we find a motion guided by the three landmarks that will bring the animat along q^2 and q^3? To keep the landmarks fixed relatively to their initial positions we connect them by three rods as shown in Figure 6. In addition, we introduce three strings qp_1, qp_2, qp_3, with weights $\alpha_1, \alpha_2, \alpha_3$. We will denote the distance between checkpoint q^i and landmark p_j as r_j^i. Since the points q^k are different positions along a single path of the animat q, we must have

$$\begin{aligned} \alpha_1 r_1^1 + \alpha_2 r_2^1 + \alpha_3 r_3^1 &= \\ \alpha_1 r_1^2 + \alpha_2 r_2^2 + \alpha_3 r_3^2 &= \\ \alpha_1 r_1^3 + \alpha_2 r_2^3 + \alpha_3 r_3^3. & \end{aligned}$$

In fact, these equations are sufficient to determine the weights α_i of the strings. For example, for the three positions as shown in Figure 6, we find that, up to a scalar multiple, the solution can be written as $(\alpha_1, \alpha_2, \alpha_3) = (0.396, -0.491, 0.268)$. Note that any scalar multiple of a solution is also a solution, since keeping $\sum \alpha_i r_i$ fixed is equivalent to keeping some multiple of it fixed. All solutions define the same path. Therefore, this flexible framework has a unique motion, containing all positions q for which the distances r_i between q and the three landmarks satisfy $0.396r_1 - 0.491r_2 + 0.268r_3 = 0.396r_1^1 - 0.491r_2^1 + 0.268r_3^1$. The motion is shown in Figure 6.

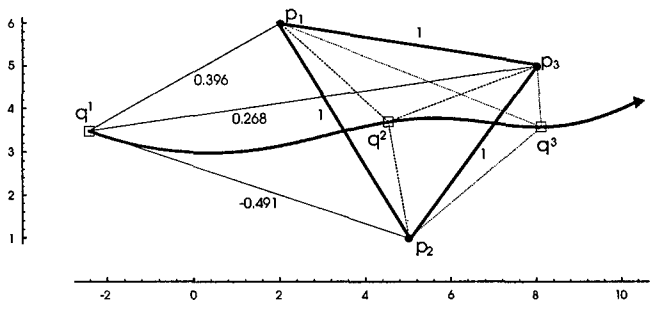


Figure 6: Framework and string weights derived from three different animat positions.

4.2 Increased adaptivity from the use of families of paths.

As we have seen animats can learn a new path from the relative distances between a number of landmarks and checkpoints. In real circumstances, however, the precise number and position of both the landmarks and checkpoints may not be known exactly, or they may be subject to continuous variation. Therefore, it is important to examine how existing paths can be rerouted if some of the landmarks are missing, and how completely new paths can be learned from old paths.

We start by noting that local path planning can be straightforwardly generalized to the case where the number of checkpoints is less than the number of landmarks. Suppose that we have three landmarks, as shown in Figure 7, but that we have only selected two checkpoints. As before, all paths passing through q^1 and q^2 must satisfy

$$\begin{aligned} \alpha_1 r_1^1 + \alpha_2 r_2^1 + \alpha_3 r_3^1 &= \\ \alpha_1 r_1^2 + \alpha_2 r_2^2 + \alpha_3 r_3^2 & \end{aligned}$$

For the positions shown in Figure 7, we now find that, up to a scalar multiple, the solutions of the above system can be written as

$$\begin{aligned} &(\alpha_1, \alpha_2, \alpha_3) = \\ &(0.326, -0.176, 0) + \beta(0.58, 0, -0.26) \end{aligned}$$

where β is an arbitrary real number. Thus we find that there is now one additional degree of freedom to choose a path. Instead of a single path, there is now a family of paths. Some of the paths in this family are shown in Figure 7. Furthermore, two paths are sufficient to represent every other path in this family. This leads to an interesting result. Suppose the animat has first learned how to travel from q^1 to q^2 while making use of the landmarks p_1 and p_2, i.e. it will have learned to keep $0.326r_1 - 0.176r_2$ fixed. Suppose in addition that it has also learned to travel along the same checkpoints q^1 to q^2, but now making use of the landmarks p_1 and p_3, i.e. keeping $0.58r_1 - 0.26r_3$ fixed. The result derived above then tells us that to follow any path from q^1 to q^2, while making use of the landmarks p_1, p_2, p_3 it is sufficient to keep some weighted sum $(0.326r_1 - 0.176r_2) + \beta(0.58r_1 - 0.26r_3)$ fixed, where β is a parameter determining which particular path is being followed. Thus, for this particular environment, by learning only two paths passing through two checkpoints, the animat has actually learned an entire family of paths passing through these two checkpoints.

Experiments with a robot. Figure 8 shows another example. Here a robot was taught to use two landmarks

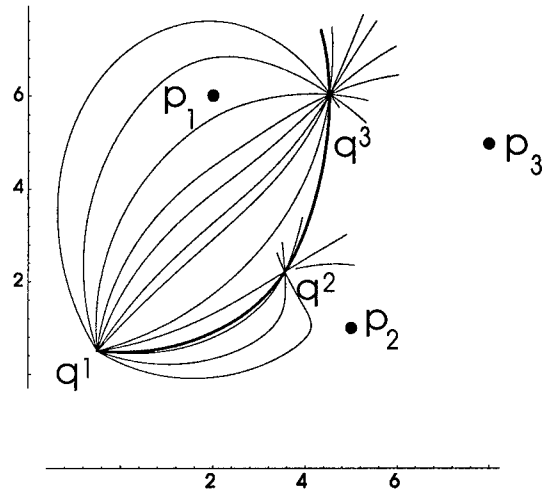


Figure 7: Animats may learn families of paths, and they may combine the families that have been learned to find a new course.

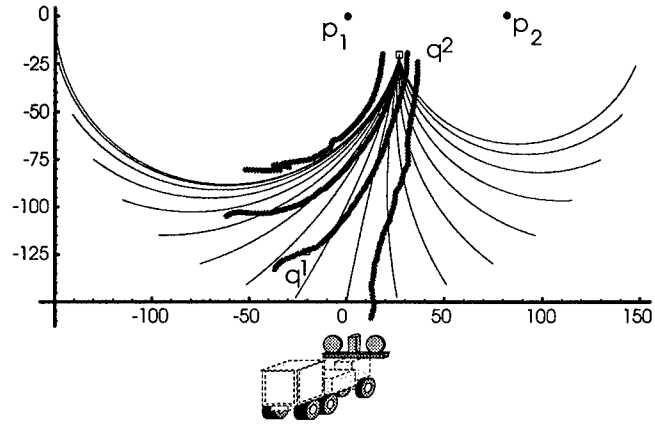


Figure 8: Environment with two landmarks, one goal, a family of paths (thin lines), and the actual robot trajectories (thick lines).

to travel to a goal, regardless of its initial position. In particular, for two landmarks p_1, p_2 we calculated the family of all paths passing through q^2. Figure 8 shows 18 different paths (thin lines) belonging to this family. Figure 8 also shows some real courses followed by a robot for some distinct initial positions. This was done according to the following principle. If we let q^1 denote the initial position of the robot, the robot first calculated a second family of paths passing through q^1. Next, the robot selected the path common to both families.

If all measurements made by the robot would have been correct, and if the mechanical control of the robot would be free of errors, then the robot would stay on the chosen path. In the real world however, the robot will deviate from its course. Therefore, at regular times, the robot recalculated its course for its new current position q^1. This guaranteed that the robot would always pass near to its goal q^2. Thus when the deviation became too large, the robot adapted its course, which can be clearly seen in Figure 8.

4.3 Learning new paths from old paths.

Suppose that, using three landmarks (Figure 7), the animat has learned all paths from q^1 to q^2 by learning at least two different paths from this class. Suppose furthermore that it has also learned all paths going from q^1 to q^3. Can the animat then use this knowledge to travel from q^2 to q^3, a route that it has never taken before? Looking carefully at what has been learned we will find that this is indeed possible. First, the animat has learned that, up to a scalar multiple, all paths going from q^1 to q^2 can be represented as

$$(\alpha_1, \alpha_2, \alpha_3) = (0.326, -0.176, 0) + \beta(0.58, 0, -0.26).$$

Similarly, it has learned that, again up to a scalar multiple, all paths going from q^1 to q^3 can be represented as

$$(\alpha_1, \alpha_2, \alpha_3) = (2.91, -0.173, 0) + \gamma(0, 0.212, -0.0176).$$

Figure 7 also shows some representatives of this second family of paths. Taking into account the scalar multiples while searching for the intersection of both families of paths, one can show that, up to a scalar multiple, the path common to both families can be represented by $(\alpha_1, \alpha_2, \alpha_3) = (0.802, 0.153, -0.488)$. This path will pass through the checkpoints of the first family of paths, as well as through the checkpoints of the second family. In particular, it will pass through q^2 and q^3, as we have asked. This path is shown in Figure 7 with a thick line.

Thus, once the animat has learned a pair of paths from q^1 to q^2, and a pair of paths from q^1 to q^3, it is possible to derive a path from q^2 to q^3 without having to learn it explicitly. It is sufficient to find a linear combination of the first pair of paths that is at the same time a linear combination of the second pair of paths.

4.4 Entering or passing-by landmark pairs

We have seen that instead of calculating a path from a fixed set of landmarks and an equal number of checkpoints, it may be more appropriate to construct new paths by combining paths that have already been learned. In this section we will look at the simplest

possible paths that can be used as building blocks to construct new paths.

If an animat wants to travel from an initial position towards a goal, and if there are no other checkpoints the animat has to pass through, two landmarks suffice to guide this motion. Thus one animat together with a pair of landmarks can be considered as the simplest non-trivial case of animat navigation. Let us now examine this situation more closely. A qualitative analysis shows that the sign of a landmark weight is more important than its absolute size. Figure 9(a) shows two landmarks p_1, p_2 with weights equal to $+1$ and -1, respectively. Whatever the initial position of the animat, it will always pass in between the two landmarks. On the other hand, if both landmarks have equal weights, then all paths will circle around both landmarks, which is illustrated in Figure 9(b).

This is no longer true for arbitrary sized weights. For example, although both weights may be positive, there may still exist paths passing in between the landmarks. Nonetheless, one can prove the following property. Figure 9(c) shows an environment with two landmarks. For each landmark there is a circle centered around it and passing through the other landmark. If the landmarks have opposite signs the animat will always pass in between the two landmarks once it is in the intersection region of both circles. Figure 9(c) shows some paths for randomly chosen weights with opposite signs, and passing in between the two landmarks. Likewise, for two landmarks that have the same sign, if the animat is outside the region bounded by either of the two circles, it will always circle around both landmarks. Figure 9(c) also shows some paths for randomly chosen weights that have the same sign. So, even for arbitrary sized weights there are important regions where only the signs of the weights matter for determining whether the path will pass in between the two landmarks or circle around them.

This simple property can often be used as a guideline for more complex situations. Even if the number of landmarks is much larger than two, the animat will often tend to pass in between landmarks that have opposite signs, and it will circle around groups of landmarks that have the same sign. In fact, as will be shown below, we can use this property for more global path planning: running through mazes.

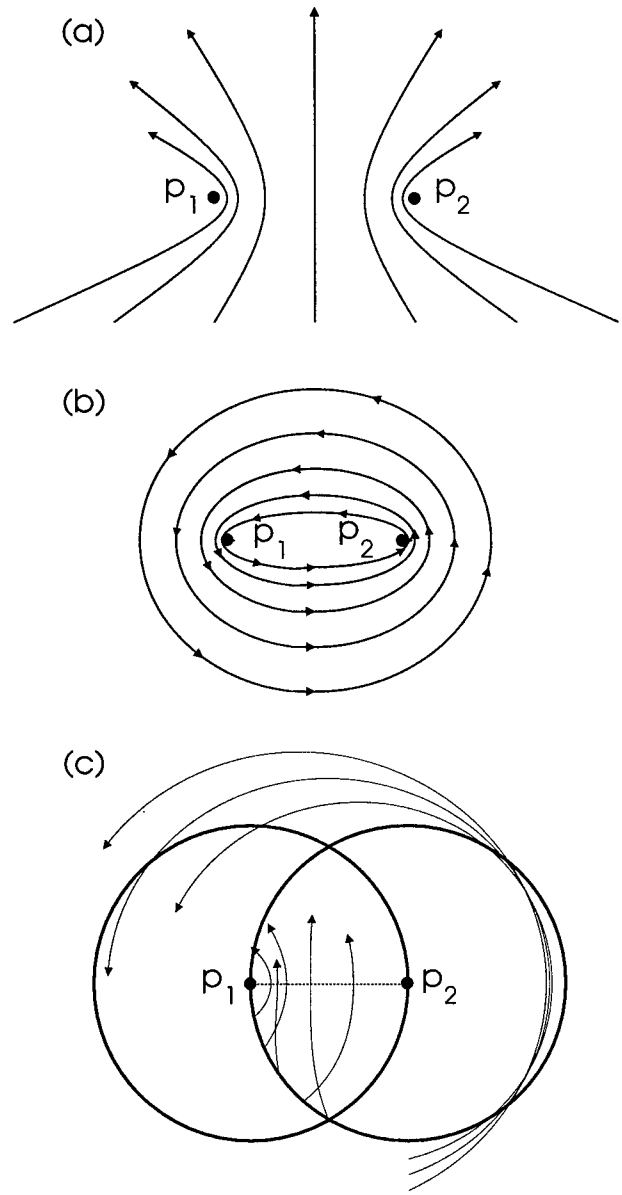


Figure 9: A qualitative analysis shows the importance of the signs of the two landmark weights.

The remarkably powerful logarithmic maze rule. Here we describe a simple example where the animat's behavior is limited to entering or passing by pairs of landmarks. Nonetheless, this approach will be shown to be powerful enough to let the animat find its way out of

a maze.

Let us consider a flexible framework where the distances between the points may also satisfy relations of the form

$$\sum_i \alpha_i \log r_i = \alpha_0.$$

Hence some of the strings in the mechanical analogue have been replaced by strings of a more sophisticated nature that somehow measure the logarithm of the distances between animats and landmarks. However, it is easy to show that to follow the paths defined by this new variant of the flexibility model the animats can use simple control laws as well. After taking derivatives we find that $\sum_i \alpha_i \dot{r}_i / r_i = 0$. It follows that in this case the velocity of the animat must satisfy

$$\frac{\dot{x}}{\dot{y}} = -\frac{\sum(\alpha_i \sin\beta_i)/r_i}{\sum(\alpha_i \cos\beta_i)/r_i}. \qquad (4)$$

Again, for each point along its course the animat can find from this simple relation the direction in which it should travel to satisfy the constraints imposed by the flexibility model. But now, the animat needs to measure, in addition to the angles of the lines of sight β_i, the relative distances r_i as well.

We introduce the logarithmic strings because, when running through mazes, logarithmic strings have certain advantages. They are very sensitive to small distances, i.e. distances for which $\log r_i$ becomes a large negative number. This is particularly useful if the animat wants to avoid collision with landmarks. Figure 10 shows the floor plan of three adjacent rooms, connected by a corridor. The landmarks in this example are no longer point-like, but wall-like instead. The distance between the animat and the wall is measured as the distance between the animat and the point closest to it on the wall. For example, the dashed line shows a set of points equidistant to one of the walls.

Figure 10 contains 14 landmarks. Four of these landmarks are walls that have received weight +1, and six landmarks are walls with weight −1. In addition there is a table in one of the rooms. Two sides of this table are landmarks with weight +1, the other sides have weight −1. As mentioned, the animat has a flexibility model using logarithmic strings, i.e. it tries to keep $\sum \alpha_i \log r_i$ fixed, where the α_i are the weights, +1 or -1, of the landmarks. Figure 10 shows some of the possible courses of the animat. The precise course depends on the initial position of the animat in the left doorway. According to what we have seen before, the animat will try to pass in between landmarks with opposite signs. In fact, all signs have been chosen in such a way that the animat will reach a given destination point at the upper side of the floor plan. Depending on its course, the animat decides to pass the table either at the left side or the right side. Hence, as indicated by this example, to find its way out of a maze an animat using a flexibility model containing logarithmic strings must learn only the weights, +1 or -1, of the landmarks it comes across or equivalently which landmarks to keep to its left and which landmarks to keep to its right.

Note that in this example we have not taken into account visibility constraints. A real animat can not see every wall included in the floor plan. However, due to the logarithmic nature of the control law, only nearby walls count, and simulations that include some visibility determining rule show that the traversed courses are changed only slightly if we impose realistic visibility constraints.

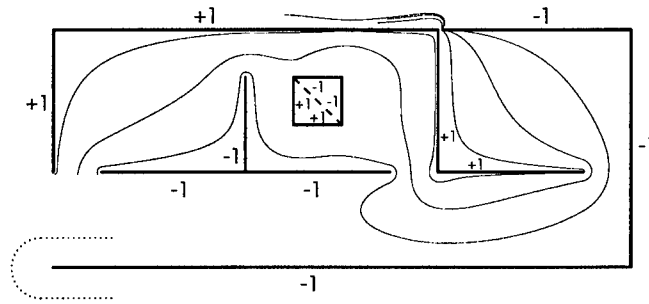


Figure 10: The logarithmic maze rule.

5 Conclusion

We have proposed a unified framework for animat navigation. Our main purpose was to define a single model that would be well suited for adaptive navigation. Therefore, we have made the assumption that it is necessary to integrate the animat's knowledge about its environment as well as its ability to plan a motion into a single model. To this end we have reformulated animat navigation as a path localization problem in such a way that it became closely related to the standard position localization problem. This approach is based on classical results in statics, i.e. the rigidity and flexibility of mechanical structures. We use the flexibility model both to verify the adequacy of the sensor data as well as to calculate the future course of the animat.

We have reached some interesting conclusions: local, landmark based, path planning can be easily accomplished; simple visual sensing, without need for stereo vision, suffices for remarkably complicated animat navigation; for path localization it suffices to measure the angles of the lines of sight of the landmarks; position lo-

calization requires one additional measurement, e.g. the distance traveled by the animat. The main conclusions, however, concern the way in which an animat can adapt its course. An animat can learn a complete family of paths by learning only a few representatives of it. It may then select the most appropriate path in this family, according to its current situation. In addition, the animat can derive an entirely new path from families of known paths, without actually having to learn the new path.

We are currently extending this work along several lines. In this paper we only consider the simplest case where there is a single animat and a set of fixed objects, which are used as landmarks. In [10] we describe how the flexibility map can be used to control groups of animats in environments consisting of both static and dynamic landmarks. Furthermore we are investigating the relation between navigation as described in this paper with navigation based on a gradient method.

Acknowledgement

This work was supported by the Interuniversity Attraction Pole No.50 initiated by the Belgian State, Prime Minister's office, Science Policy Programme, and as a research project (BOZF96-18) of the University of Ghent.

References

[1] R. Arkin. Motor schema-based mobile robot navigation. *International Journal of Robotics Research*, 8(4):92–112, August 1989.

[2] L. Asimow and B. Roth. The rigidity of graphs. *Trans. Amer. Math. Soc.* , 245:279–289, 1978.

[3] Ed. by J. Paillard. *Brain and Space.* Oxford University Press, Oxford, 1991.

[4] Ed. by P. Ellen and C. Thinus-Blanc. *Cognitive Processes and Spatial Orientation in Animal and Man*, volume 36 and 37 of *NATO ASI Series D: Behavioural and Social Sciences.* Martinus Nijhoff Publishers, Dordrecht, 1987.

[5] R. Connelly. Rigidity and energy. *Invent. Math.*, 66:11–33, 1982.

[6] B. Kuipers and Y. Byun. A qualitative approach to robot exploration and map-learning. In *Proc. of the AAAI sponsored 1987 Workshop on Spatial Reasoning and Multi-sensor Fusion*, pages 390–404, St. Charles, October 1987.

[7] M. Mataric. Integration of representation into goal-driven behavior-based robots. *IEEE Transactions on Robotics and Automation*, 8(3):304–312, June 1992.

[8] J. McClelland, D. Rumelhart, and the PDP Research Group. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 2, chapter 23. The MIT Press, Cambridge, 1987.

[9] A. Recksi. *Matroid Theory and its Applications in Electric Network Theory and in Statics.* Springer-Verlag, Berlin, 1989.

[10] P. Veelaert and H. Peremans. Flexibility models for robot navigation. Technical Report DG96-02, University of Ghent, Department ELIS, 1996.

[11] T. H. Waterman. *Animal Navigation.* Scientific American Library. W.H. Freeman and Company, New York, 1989.

Maze Navigation Using Optical Flow

Andrew P. Duchon

Department of Cognitive and Linguistic Sciences
Brown University
190 Thayer St.
Providence, RI 02912-1978
email: duchon@cog.brown.edu

Abstract

Some recent work with autonomous robots has focused on using optical flow for "direct" control of speed and rotation in obstacle avoidance and other simple behaviors. This work has been inspired by work with insects showing similar mechanisms. To extend these behaviors, three methods of maze navigation are investigated in a simulated robot modeled after a real one. A motor-based method places biases in the obstacle avoidance control law used previously. A perception-based method uses optical flow to detect possibilities for action (e.g., to turn left or right). Both of these require that the agent have a list of biases in order to navigate. The third method, called the Salience Centroid Method, is based on a theory of the role of the hippocampus in rat navigation. This method trades off the memory of the first two for more advanced perceptual processing and allows the most flexible behavior.

1 Introduction

In the last few years, a number of researchers have explored the use of optical flow for directly controlling a robot's behavior (Coombs & Roberts, 1993; Duchon & Warren, 1994; Sobey, 1994; Santos-Victor et al., 1995; Coombs et al., 1995). This research has concentrated on methods for obstacle avoidance without modeling the world (Aloimonos, 1992) and has even produced robots which can dock (Santos-Victor & Sandini, 1994) and play "tag" (Duchon et al., 1995). These implementations on a robot platform have been inspired by theories of how bees navigate through the world (e.g., Srinivasan, 1992). The crossover between these two fields indicates that it is possible to speak in general terms of an *agent*. Results in both fields apply to any agent, be it biological or artificial, that can detect optical flow and act upon it.

This paper begins two extensions to the insect-based, on-line steering work cited above. The first is to develop methods using optical flow for goal-directed behaviors, still without using the information in optical flow for creating a 3-D model of the environment. The second purpose is to take inspiration from another animal literature, namely cognitive mapping in rats. As a first pass at these higher-level behaviors we use a simulated agent which is modeled after the real robot and "control laws" presented previously (Duchon & Warren, 1994; Duchon et al., 1995).

Three strategies for maze navigation are explored. The "motor-based strategy" is similar to the kinesthetic response chains of the early rat-in-a-maze literature whereby "the act of running a maze seemed to be more a series of concatenated reflex movements released by kinesthetic cues" (Munn, 1950, p.186). This conclusion was dismissed with further experimentation (e.g., Dashiell, 1930) and the simulation also quickly shows the limitations of this strategy. The "perception-based strategy" takes more advantage of the information in optical flow and allows the agent to make choices and learn a route-based representation of the environment. Although much more flexible, this strategy too has its faults. Finally, we explore the use of optical flow in a modification of O'Keefe's (1989, 1991) model of hippocampal function. This method alone also has its problems which are addressed in the conclusion.

2 The Simulated Environment

2.1 The environment and the agent

The maze environment (Figure 1) has many possibilities for action including cul-de-sacs, and T-, L-, and U-junctions. The agent is an observation point with a

field of view (FOV) of $0° - 330°$, and although a point, a triangle is drawn around it for display purposes. For obstacle avoidance and for the first two strategies described below, the agent has 48 samples from a FOV of 120°. The agent has access only to the optical flow "seen" at the intersection of each sampled line of sight and the nearest maze wall.

2.2 Calculation of the optical flow

The instantaneous optical flow from translation only is used. No rotational components are added (a condition our robot approximates (Duchon & Warren, 1994) since it moves in a piecewise linear manner). As such, the optical flow can be calculated analytically. Because the simulations take place in a maze without other agents, the optical flow value (here, angular velocity in the horizontal dimension only) for a given line of sight is simply

$$\dot{\beta} = \frac{|\mathbf{h}| \sin\beta}{d} \tag{1}$$

where $\dot{\beta}$ is the angular velocity, $\mathbf{h}$ is the velocity of the agent (here, the agent always moves along the optical axis), β is the angle of the sample with respect to $\mathbf{h}$, and d is the distance between the observation point and the sampled point on the wall. In this manner, the observer can obtain a speed-scaled distance between itself and the sampled points in the environment. Another piece of information provided by optical flow is the time-to-contact, known as τ (Lee, 1976). For small values of β, $\tau \approx \beta/\dot{\beta}$. In the maze, a low τ value signals that there is a wall in front of the agent. If τ is below a threshold, the agent will turn 180° (the "tau-reflex").

2.3 Obstacle Avoidance

For obstacle avoidance, we used the Balance (or Centering) Strategy which has proven useful in robot implementations (Duchon & Warren, 1994; Coombs et al., 1995; Santos-Victor et al., 1995). With the Balance Strategy, the agent moves so as to equate the average magnitude of flow detected on each side of the optical axis (which is tied to the heading direction both in simulation and in our robot). At each time step in the simulation, the instantaneous optical flow values are obtained for each sample. The average for each side ($\overline{\dot{\beta}_L}$, $\overline{\dot{\beta}_R}$) is calculated and the agent rotates an angle r as given by

$$r_{bal} = k(\overline{\dot{\beta}_L} - \overline{\dot{\beta}_R}) \tag{2}$$

where k is a scaling constant. With $r_{bal} > 0$ indicating clockwise rotation, more flow on the left causes the agent to turn to the right.

The essential idea behind this strategy is that of motion parallax: when the agent is translating, closer objects will give rise to greater motion across the retina than farther objects. It also takes advantage of perspective in that closer objects will also take up more of the field of view, biasing the average towards their associated flow. In the end, the agent turns away from the side of greater flow, but only such that it does not turn into an obstacle on the other side. When the agent is in a hallway, it will tend to move down the center. Obstacle avoidance using the Balance Strategy has proven to be very robust with robots, but the question remains as to how to integrate this behavior with more goal-directed actions. Below, we explore a few possibilities.

3 A Motor-based Method

It is possible to induce other forms of behavior by placing biases in the Balance Strategy:

$$r_{bias}(\gamma) = k[(1+\gamma)\overline{\dot{\beta}_L} - (1-\gamma)\overline{\dot{\beta}_R}] \tag{3}$$

Since $r > 0$ indicates a right turn, a positive bias ($\gamma > 0$) makes the agent hug the right wall. With a constant bias, this control law will lead to wall-following, which can be used to map an environment (Mataric, 1992). With a variable bias however, say of a given sequence, the agent can navigate through the maze to a pre-determined location. We implemented this latter strategy.

Let's posit a given sequence of biases (from some previous experience) which we know will lead the agent to the goal location, given that it always starts at the same start location (*home*). The agent will need to both make turns and be able to pass by openings that are undesired. To pass an opening on the left, it can use a small positive bias (to the right) that is enough to keep it from going into the left opening. To make a turn into an opening, a stronger bias is applied. For example, the sequence of biases used in Figure 1 is : Γ = {PASSRIGHT, TURNLEFT, PASSRIGHT, TURNRIGHT, CUL-DE-SAC, STOP} where TURNLEFT = -0.3, TURNRIGHT = 0.3, PASSLEFT = 0.15, PASSRIGHT = -0.15, and CUL-DE-SAC = 0.0.

Since the agent only has access to the sequence of biases, it must have a way of determining whether or not it has actually made the turn or passed by the opening. This is done by keeping a moving average of the last few r values, denoted by σ, an "efference copy" of its movement. If it is not currently in a turn but the pointer in the sequence indicates the next action is a turn, then $|\sigma|$ moving above a threshold, θ_t, indicates it has entered the opening. When $|\sigma|$ again falls below θ_t, it has completed the turn. Similarly, passage by an opening can be detected as σ rises above and falls below another threshold, θ_p, characteristic of a slight turn into an opening and a straightening in the next

passage. A cul-de-sac can be detected by a low τ value which will also trigger the tau-reflex. Each time the agent completes a turn, pass or cul-de-sac, a pointer moves to the next bias in the list.

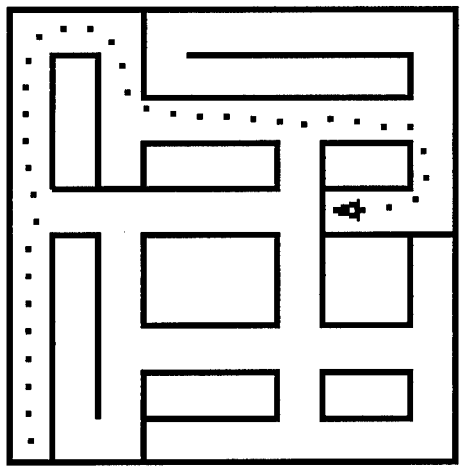


Figure 1: *Path of the agent using the motor-based method and the sequence in the text.* The agent starts in the lower left cul-de-sac. The dots indicate the agent's location every 20 time steps. Notice that it does not travel down the middle of the passageways, but rather hugs the walls depending on the bias γ. For example, the first dot is in the middle of the passage, but the agent moves a bit to the left in order to pass by the opening on the right. It "dips" into the passage only slightly, the recovery from which signals that it has passed the opening so it moves to the next bias in the list.

There are a number of problems with this method of navigation. First and foremost, the agent must know the sequence beforehand and how this comes about is undetermined. Second, on the motor side there is a process that indicates it has completed a turn or passed by an opening, but there is nothing on the perceptual side by which the agent can "see" that it has a choice. To address these two issues we implemented a more perception-based method for navigating the maze.

4 A Perception-based Method

Given the rectilinear nature of the maze, in which all walls come together at right angles, the agent could "see" that there is an opening on the left or right by detecting a large drop in $\dot{\beta}$ from a given sample in the periphery to the next, more central, sample. An opening straight ahead is given by the central τ value. If τ is low enough, then a wall must be in front of the agent, if high enough, an opening. The agent must simply choose which among the possibilities (go straight at a cross-junction; turn or pass left or right at a T-junction) it will take and apply the appropriate bias in Equation 3. In the motor-based method, the current bias in the list was applied constantly. With this method, the bias is applied only when at least two possibilities are available (a *choice point*) and only for a limited period (otherwise, the bias is zero). The latter condition is applicable because the agent will typically detect the opening at a fixed point in time before it must turn. Thus there is no need for an efference copy with this method.

The basic task for the agent is given that it starts in a cul-de-sac (*home*), it will move through the maze until it reaches another cul-de-sac, at which point it returns home (Figure 2). On the way out, at each choice point the agent makes a random choice among the possibilities and records the choice in a list. If, say, there is an opening to the right only, and a wall is in front of the agent, then there is only one way to go and no need to make or record a choice. Again, the list consists of a series of biases (Γ). For the return trip, the agent simply goes backwards through the list. At each choice point, the agent applies the negative of the current bias in the list and moves the pointer back one.

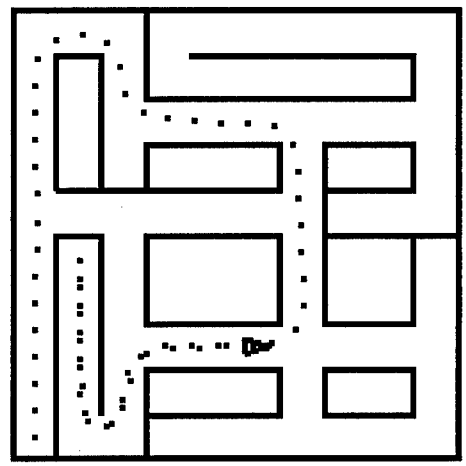


Figure 2: *Perception-based maze behavior.* The agent again starts in the lower left corner. It stays in the middle of the passage because there is no bias until it reaches a choice point halfway up the left side. There is a slight deflection at the first passage despite the PASSRIGHT bias. At the frame shown, it has made its way to the cul-de-sac at the middle left side and begun the trip home (doubled-up dots). It can consistently find its way back from any cul-de-sac.

There are a number of disadvantages to this method as well. First, as noted above, this method depends on the regularity of the maze. For instance, to determine if there is an opening, at least a 35% change in $\dot{\beta}$ is required in one of the peripheral samples given a 120° FOV. There is some flexibility in these values (in the percent change and the FOV), but not much. It also

depends on the size of the passageways and doorways and there is no contingency for entering a room.

Second, it is a route-based navigation system which means it is very "fragile" (O'Keefe & Nadel, 1978). The exact route taken on the way out must be used on the way back; no shortcuts can be taken. Moreover, if the agent is heading home and gets transported somewhere else in the maze, its list of biases is useless. A solution to this problem would be to make a connectivity graph of the possibilities at each junction and the choice made. The entire maze could then be represented in this way and standard AI techniques could be used to determine, say, the sequence of biases (a policy) to be applied at each choice point in order to get home from any location in the maze.

Third, a representation in terms of a list (or graph) is unsatisfying. It implies a symbolic memory mechanism, which behavior-based robotics is trying to avoid (however, see Mataric, 1992). Ideally, there would be a perception-based method, with minimum memory, which would use "higher-order invariants" in the environment for more flexible navigation.

5 The Salience Centroid Method

In many of the studies of maze-learning in rats, placed above the maze are a number of cues which can be seen at all times, termed "extra-maze cues" (Munn, 1950, p. 192). Most of these studies have shown that if the extra-maze cues are rotated a certain angle after the rat has learned where the goal is, the rat will now behave as if the goal location had also rotated by the same angle. That is, it is behaving with respect to a frame of reference defined by the extra-maze cues. This is apparent not only behaviorally, but also physiologically. It is now well known (O'Keefe & Nadel, 1978) that certain cells ("place cells") in the hippocampus are active only when a rat is in a certain location in its environment (the "place field"). The place field of a cell also rotates with the cues (Muller & Kubie, 1987).

O'Keefe (1989) proposed that the hippocampal formation performs matrix transformations in a Cartesian mapping system. In his model, a current matrix $\mathbf{C}$ contains the *egocentric* (x, y) coordinates of every object (as obtained by converting the distance and bearing of each object). Going to a goal requires multiplying the inverse of the current matrix by the goal matrix $\mathbf{G}$ to get the transform, or movement, required: $\mathbf{C}^{-1} \cdot \mathbf{G} = \mathbf{T}$. O'Keefe proposed how various mechanisms in the hippocampal formation could carry out these operations, such that given $\mathbf{C}$ (from vision) and $\mathbf{G}$ (from previous experience), the hippocampus could generate the correct movement $\mathbf{T}$ to move the rat to the goal.

In his second model (O'Keefe, 1991), coordinates are no longer egocentric, but make use of a "centroid-slope" construction. Once the bearing and distance of all the objects are obtained, the *centroid* is simply the average of the vectors from the rat to each object. The centroid is invariant to rotation, thus gives a radial distance for the rat's location. The *slope* (for two objects) is simply the ratio of the difference of the y- to the x-coordinates, also in an egocentric frame. The slope is invariant with respect to translation, and specifies the rat's orientation. The distance from the centroid and the angle from the slope are both invariant as long as the same objects can be seen. Movements are calculated with respect to these values, and not to egocentric coordinates as in the first model. However, one problem with this method is that the symmetry on the two sides of the "slope" means there is always an ambiguity in the rat's location. Another issue is the computational expense of averaging the slope of *all* pairs of cues.

The method below is proposed for three reasons. First, it resolves the symmetry problem and scales linearly with the number of cues. Second, it exploits the information in optical flow (as O'Keefe (1991) actually suggests). And third, this method, like the previous two, integrates goal-directed movements with obstacle avoidance.

Three assumptions are required for the method to work. The first is that all the extra-maze cues must be within view simultaneously for the agent to "fix" its position. This "omnipresence" of the cues is assumed by most current computational models of the hippocampus since nearly all physiological studies have recorded place cell activity in an environment where all the cues could be seen[1]. The second assumption is that at least one of the extra-maze cues must be perceptually distinguishable from the others. The third assumption is that the agent has some sense of its rate of motion, if only proportionally. Reasons for these assumptions will become apparent and methods for removing the first one will be discussed below.

To comply with these assumptions, the environment and the agent must be slightly modified. The environment is now equipped with four "extra-maze cues" which can be seen from anywhere in the maze. Each cue is assigned a random number as its "salience." The agent now has a second set of optical flow samples, different from those described above which are still used

[1] Once the rat is placed in an environment, place cell activity is maintained even as cues are taken away (O'Keefe & Speakman, 1987) or the lights are turned out (Quirk et al., 1990). This ability to use dead-reckoning to maintain place cell properties is mediated mostly by vestibular information modified by optical flow (Sharp et al., 1995). No dead-reckoning is used in any of the methods presented here.

for obstacle avoidance. This second set can be thought of as another set of horizontal motion detectors, but now in a plane above the rat. Thus, they only detect the extra-maze cues. Forty-eight samples are taken from a 330° FOV.

5.1 Salience Centroid Algorithm

One flow value $\dot{\beta}$ at an edge of each extra-maze cue is detected (an edge is detected simply by a large difference between two adjacent samples). The (speed-scaled) distance to each cue is $d = \sin(|\beta|)/\dot{\beta}$. For the purposes of computation only, we convert these (d, β) values to egocentric (x, y) coordinates (the y axis being the optical axis of the agent). The coordinates of the "centroid" (c) then, is simply the average of the (x, y) coordinates of the four cues: $(\bar{x}, \bar{y})$. A directed vector (as opposed to just a slope) is required for an unambiguous coordinate system of the environment. Therefore, the original x and y values of each extra-maze cue are multiplied by the "salience" of the cue and a new average obtained. At least one of the cues must have a different salience than the others for the "salience centroid" (s) to be different from c.

Given s and c, the salience vector **s** can be found (see Figure 3A). Through a couple of coordinate transforms, the location of the observer with respect to **s** can be calculated, now back in polar coordinates: (r_O, θ_O). We should note that crucial to this calculation is ψ, the bearing of c with respect to the agent's heading (**h**), as well as ω, the agent's heading (**h**) with respect to (**s**). A little more vector algebra (Figure 3B) will determine the bearing β, with respect to the current heading, of a known location in the environment (r_G, θ_G).

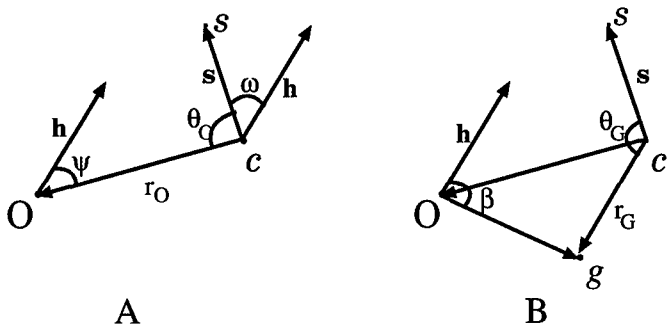


Figure 3: *Vectors used in the calculations for the Salience Centroid Method.* See text.

5.2 Using the Salience Centroid

At any given moment when all the extra-maze cues can be seen, the agent knows its location and the bearing (β) of the goal point (if there is one). However, the agent need not turn instantly towards the goal. Instead, a certain percentage, $perc$=0.2, of β is used.

A few other factors need to be considered as well. The first is that if the goal is on the other side of a wall from the agent, the agent should not beat its head against the wall, as it were. This can be avoided by taking the total sum of optical flow that the agent sees (Σ), and reducing the bias accordingly:

$$\gamma = \beta \times perc \times (1 - \frac{\Sigma}{a}) \qquad (4)$$

where a is an appropriately scaled constant. Other optic variables, like τ, could be used, but the sum takes into account the whole field of view in an appropriate and general way (Figure 4).

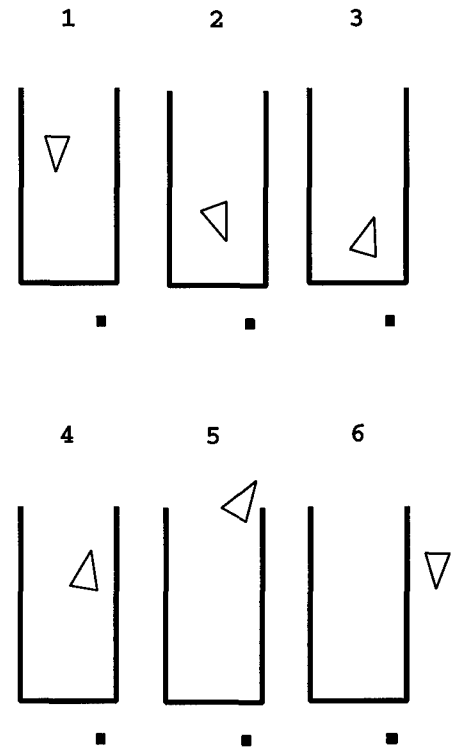


Figure 4: *Escaping from a "goal-directed blind."* The sequence shows the agent (1) entering a "box" because the goal (square) is on the other side. As it nears the end (2), Σ increases, which reduces γ and so the Balance Strategy predominates. This allows the agent to turn itself around (3) and begin going up the wall (4) since being near the wall, Σ stays high. Without using Σ, the agent would stay in the cul-de-sac, continually using the tau-reflex to flip around and the bias to pull it back in. Once the agent gets to the corner (5), Σ becomes low, γ becomes high, and the agent begins heading towards the goal again (6).

The bias (γ) here is a function of the direction of the goal and is given in degrees. In the other two methods, it was related to an event (turn left or right) and did

not have a specified relation to the desired amount of rotation. Therefore, although this bias may still work in Equation 3, in these simulations it is added directly to the values from Equation 2. That is,

$$r_{out} = r_{bal} + \gamma \quad (5)$$

Another reason for not using this γ in Equation 3 is that it comes from a different source: the set of optical flow samples from "above" the maze. In addition, the previous methods relied on the regular structure of the environment to achieve their ends. However, this method makes no assumptions about the shape of the environment. When walls are far away from the agent (e.g., at a choice point), γ will be predominate in the agent's behavior; but when the walls are close, r_{bal} will have more of an influence.

Another factor to be considered is the speed of the agent, which has two effects. First, the entire arrangement is scaled to the agent's speed. But no metric values are required, only proportional ones. That is, the current r_O and the goal's r_G, could be multiplied by some speed factor, ρ, which reflects the agent's current speed proportional to all others[2]. So, during normal cruising, $\rho = 1$. The only reason (in these simulations at least) to go at $\rho < 1$ is when nearing a goal that is next to a wall. Because Σ reduces the influence of the goal on the agent's turning, the agent will not be able to approach goals near walls. But, by reducing the agent's speed when it is close to the goal ($\rho = 0.5$), Σ is also reduced and the agent can reach the goal.

In our implementation, the agent wandered through the maze until a button was pressed: the current location was logged as (r_G, θ_G) (where $r = r \times \rho$) and the agent was transported to a random position and orientation elsewhere in the maze. Thus, the only memory required was that of the two coordinates of the goal which came from a place it had already been. All the other information was captured on-line. The agent had a 330° FOV for detecting the cues. If not all of them could be seen, then $\beta = 0$. The samples used for input to the Balance Strategy, however, only came from the center 120°. Finally, in our simulations, the goal itself was invisible: the agent reached it when it was at an arbitrarily close distance (similar to a Morris (1981) water maze where the rat must swim to an unseen platform just under the surface of milky water). In more realistic circumstances, all that is needed is that the Salience Centroid Method bring the agent to within view of the goal.

[2]This proportion could be detected either proprioceptively through the joints, or visually, e.g., by the optical flow from the ground plane.

5.3 Results and discussion

Figure 5 shows an example of the method at work. Although the agent makes a few wrong turns into goal-directed blinds (a difficulty well-known in the rat literature, e.g., Buel, 1935), it does find its way to goal.

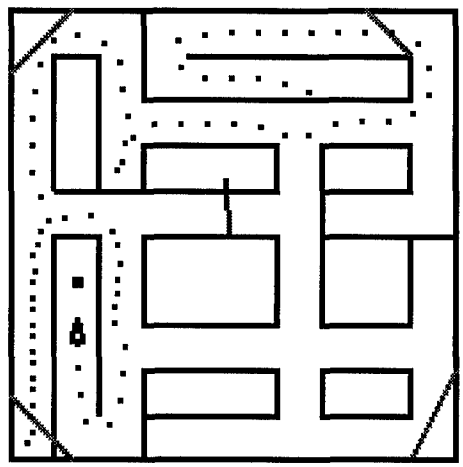


Figure 5: *Path of the agent using the Centroid-Salience Method.* The diagonal gray lines in the corners represent the "extra-maze cues" which can be seen at all times. The line in the middle represents s and points upwards in the diagram. After wandering through the maze (not shown), the agent ended up in the cul-de-sac in the lower left corner (solid square) and was transported to the upper right (where the dots begin). It made correct decisions until it reached the small alley in the upper left which was goalwards. However, it left the cul-de-sac by means of the tau-reflex and continued around to its left. When it reached the intersection at the middle left it turned to the left slightly, but because the goal was to the right of the corner, it veered right, down the cul-de-sac. Coming back from the cul-de-sac it wound its way around to the goal.

One unfortunate effect of the C-S Method is shown in Figure 6 *Top* where the agent starts off with the goal to its left, so it makes a left turn. This however leads it in an endless loop since it will always end up back in this same position. Other forms of information, like the perception-based method above, would be required to detect this kind of circumstance and change the agent's behavior: all that is required is a full turnabout anywhere on the path; moving in the clockwise direction will allow the agent to reach the goal. Another way to avoid this situation is to add noise to the system (Figure 6 *Bottom*), such that on average it would go to the left, but would occasionally go to the right at the crucial intersection and reach the goal.

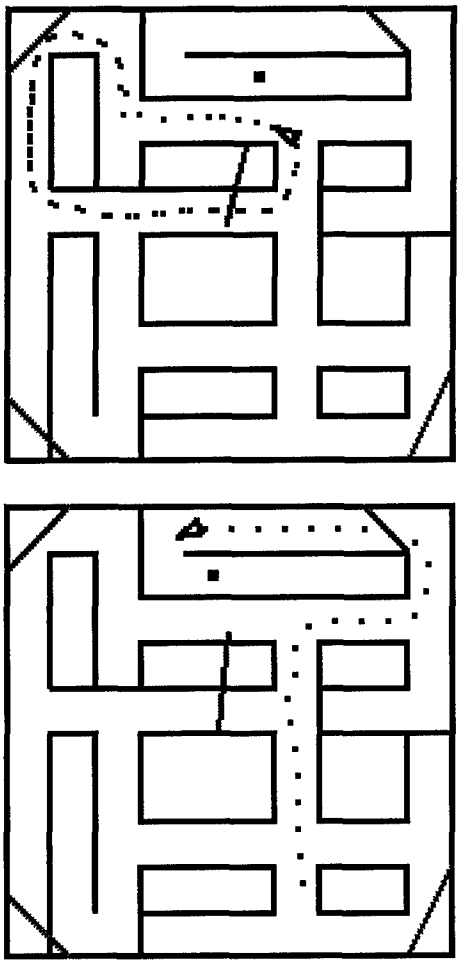


Figure 6: *Top*: The agent is caught in a loop (a local minimum) while trying to reach the goal. *Bottom*: With small Gaussian noise added to the observed flow values, the agent, by chance, makes a right turn at the intersection and can reach the goal.

5.4 Physiological Correlates

O'Keefe (1989, 1991) details much of the hippocampal physiology necessary to carry out the operations which we have proposed (which are in fact similar to his, but are based explicitly on optical flow). In brief, the theta rhythm in the hippocampus can be used to convert between polar and Cartesian coordinates (see O'Keefe & Recce (1993) for other use of the theta rhythm and a recent re-working of O'Keefe's model). The place cells themselves would represent (r_O, θ_O). Hippocampal place cells are sensitive to speed, either being modulated or tuned by speed (Wiener et al., 1989; Mizumori et al., 1990), and become nearly silent when the rat is not moving which may reflect the influence of ρ. Some cells in the postsubiculum are sensitive to head direction (Taube et al., 1990) and might represent ω, the angle between the heading and the salience vector. Cells in the hippocampus proper are also sensitive to the direction of motion through the place field (McNaughton et al., 1983) and might be related to modulaton by ψ, the angle of the heading with respect to the centroid. Or, they could be related to β, the direction of the goal. Recent evidence supports this idea. Markus et al. (1995) demonstrated that changes in task, i.e., changes in goal locations, resulted in changes in the directional selectivity of hippocampal place cells.

6 Conclusion

We have described a progression of methods an agent could use to navigate a maze. The most direct method was to place a bias in the control law for obstacle avoidance, but this required giving the agent a predetermined list of biases. The second method depended on the agent being able to detect openings, make a choice among them, record the choice (as a bias), and backtrack through the list of choices in order to return home. As mentioned above, this method could be used to create a connectivity graph of the environment. The final method we described required a nearly continuous perception of the agent's place within the environment by means of which it could find the heading direction of another (goal) location. So, although it needed more on-line processing (a separate system to determine the agent's relative location), memory was reduced to two numbers which could easily be considered as a kind of set-point for an error reduction system.

One might suspect that a memory system would also be required for the landmarks to be recognized, but this is not necessarily true. All that is needed is that the objects have the same "salience" ("quality" in O'Keefe, 1989) which is consistent across viewpoints. We have left "salience" essentially undefined for it could be any number of perceptual properties, or, more likely, a conglomerate of all of them. One intriguing idea is that a particular environment could be characterized by the length of the salience vector—a single value for which there is little chance of close matches among different environments.

One of the assumptions for the Salience Centroid Method was that all the cues be available at all times. This is true of most current models (e.g., O'Keefe (1991), Bachelder & Waxman (1994), Shapiro & Hetherington (1993); but see Touretzky et al. (1994), Worden (1992)). One of the properties of the hippocampus which is prominent in most models of its function is its autoassociativity. This is basic to many models of memory and allows for a "whole" memory to be recreated from a part. We have begun to look at using the autoassociative BSB-model (Brain-State-in-a-Box, see

e.g., Anderson, 1995, pp. 493-628) as a model of the hippocampus and hope to incorporate this work into our simulations. Such a model would allow the agent to determine the salience vector by only seeing a few of the objects. Once given this capability, it would not be difficult to "string along" a series of salience vectors, or create an extended one, which connected a number of distinct environments.

Moreover, the BSB network would be part of the perceptual process, which, if motor commands are also included, would create a single integrated system for both perception, action, and representations of the environment. That is, the agent's actual perception of the environment would change as a function of learning in the BSB, which would change its behavior, which would change what it learned, etc. This would be an example of what Gibson called learning through differentiation (Gibson, 1966, p. 270). A number of criticisms of the ecological approach refer to the problems it has in explaining knowledge of things that are not currently perceivable. The autoassociative networks we have begun to work with indicate a way for that which can be perceived to *specify for the agent* that which cannot be seen. Once such a system is developed in simulation we will bring these methods back to the robots with which our work began.

7 Acknowledgments

This work was supported by a National Science Foundation Graduate Research Fellowship. Thanks to Leslie Pack Kaelbling for reading an earlier draft of this paper.

References

Aloimonos, Y. (1992). Is visual reconstruction necessary? Obstacle avoidance without passive ranging. *Journal of Robotic Systems, 9*(6), 843–858.

Anderson, J. A. (1995). *An Introduction to Neural Networks.* Cambridge, MA: MIT Press.

Bachelder, I. A. & Waxman, A. M. (1994). Mobile robot visual mapping and localization: A view-based neurocomputational architecture that emulates hippocampal place learning. *Neural Networks, 7*(6/7), 1083–1099.

Buel, J. (1935). Differential errors in animal mazes. *Psychological Bulletin, 32*, 67–99.

Coombs, D. & Roberts, K. (1993). Centering behavior using peripheral vision. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 440–445 New York.

Coombs, D., Herman, M., Hong, T., & Nashman, M. (1995). Real-time obstacle avoidance using central flow divergence and peripheral flow. In *Prodeedings of the 5th International Conference on Computer Vision.*

Dashiell, J. F. (1930). Direction orientation in maze running by the white rat. *Comparative Psychology Monographs, 7*, 72 pp.

Duchon, A. P. & Warren, W. H. (1994). Robot navigation from a Gibsonian viewpoint. In *Proceedings of the IEEE International Conference on Sytems, Man, and Cybernetics*, pp. 2272–2277 Piscataway, NJ. IEEE.

Duchon, A. P., Warren, W. H., & Kaelbling, L. P. (1995). Ecological robotics: Controlling behavior with optical flow. In J. D. Moore & J. F. Lehman (Eds.), *Proceedings of the 17th Annual Conference of the Cognitive Science Society*, pp. 164–169 Mahwah, NJ. Lawrence Erlbaum Associates.

Gibson, J. J. (1966). *The Senses Considered as Perceptual Systems.* Boston: Houghton Mifflin.

Lee, D. N. (1976). A theory of visual control of braking based on information about time-to-collision. *Perception, 5*, 437–459.

Markus, E. J., Qin, Y.-L., Leonard, B., Skaggs, W. E., McNaughton, B. L., & Barnes, C. A. (1995). Interactions between location and task affect the spatial and directional firing of hippocampal neurons. *The Journal of Neuroscience, 15*(11), 7079–7094.

Mataric, M. (1992). Integration of representation into goal-driven behavior-based robots. *IEEE Transactions on Robotics and Automation, 8*(3), 304–312.

McNaughton, B. L., Barnes, C. A., & O'Keefe, J. (1983). The contributions of position, direction, and velocity to single-unit activity in the hippocampus of freely-moving rats. *Experimental Brain Research, 52*, 41–49.

Mizumori, S. J. Y., Barnes, C. A., & McNaughton, B. L. (1990). Behavioral correlates of theta-on and theta-off cells recorded from hippocampal formation of mature young and aged rats. *Experimental Brain Research, 80*, 365–373.

Morris, R. G. M. (1981). Spatial localization does not require the presence of local cues. *Learning and Motivation, 12*, 239–260.

Muller, R. U. & Kubie, J. L. (1987). The effects of changes in the environment on the spatial firing of

hippocampal complex-spike cells. *The Journal of Neuroscience*, *7*(7), 1951–1968.

Munn, N. L. (1950). *Handbook of Psychological Research on the Rat.* Boston: Houghton Mifflin.

O'Keefe, J. (1989). Computations the hippocampus might perform. In L. Nadel, L. A. Cooper, P. Culicover, & R. M. Harnish (Eds.), *Neural Connections, Mental Computations*, pp. 225–284. Cambridge, MA: MIT Press.

O'Keefe, J. (1991). The hippocampal cognitive map and navigational strategies. In J. Paillard (Ed.), *Brain and Space*, pp. 273–295. Oxford: Oxford University Press.

O'Keefe, J. & Recce, M. (1993). Phase relationship between hippocampal place units and the eeg theta rhythm. *Hippocampus*, *3*(3), 317–330.

O'Keefe, J. & Speakman, A. (1987). Single unit activity in the rat hippocampus during a spatial memory task. *Experimental Brain Research*, *68*, 1–27.

O'Keefe, J. & Nadel, L. (1978). *The Hippocampus as a Cognitive Map.* Oxford: Clarendon Press.

Quirk, G. J., Muller, R. U., & Kubie, J. L. (1990). The firing of hippocampal place cells in the dark depends on the rat's recent experience. *The Journal of Neuroscience*, *10*(6), 2008–2017.

Santos-Victor, J. & Sandini, F. (1994). Visual behaviors for docking. Tech. rep. TR 2/94, LIRA-Lab: DIST University of Genoa.

Santos-Victor, J., Sandini, G., Curotto, F., & Garibaldi, S. (1995). Divergent stereo in autonomous navigation: From bees to robot. *International Journal of Computer Vision*, *14*, 159–177.

Shapiro, M. L. & Hetherington, P. A. (1993). A simple network model simulates hippocampal place fields: Parametric analyses and physiological predictions. *Behavioral Neuroscience*, *107*(1), 34–50.

Sharp, P. E., Blair, H. T., Etkin, D., & Tzanetos, D. B. (1995). Influences of vestibular and visual motion information on the spatial firing patterns of hippocampal place cells. *The Journal of Neuroscience*, *15*(1), 173–189.

Sobey, P. (1994). Active navigation with a monocular robot. *Biological Cybernetics*, *71*, 433–440.

Srinivasan, M. V. (1992). How bees exploit optic flow: Behavioural experiments and neural models. *Philosophical Transactions of the Royal Society of London B*, *337*, 253–259.

Taube, J. S., Muller, R. U., & Ranck, J. B. (1990). Head-direction cells recorded from the postsubiculum in freely moving rats. I. Description and quantitative analysis. *The Journal of Neuroscience*, *10*(2), 420–435.

Touretzky, D. S., Wan, H. S., & Redish, A. D. (1994). Neural representation of space in rats and robots. In J. M. Zurada & R. J. Marks (Eds.), *Computational Intelligence: Imitating Life: Proceedings of the symposium held at the 1994 IEEE World Congress on Computational Intelligence.* IEEE Press.

Wiener, S. I., Paul, C. A., & Eichenbaum, H. (1989). Spatial and behavioral correlates of hippocampal neuronal activity. *The Journal of Neuroscience*, *9*(8), 2737–2763.

Worden, R. (1992). Navigation by fragment fitting: A theory of hippocampal function. *Hippocampus*, *2*(2), 165–188.

An Autonomous System for Extracting Fuzzy Behavioral Rules in Mobile Robotics

A. G. Pipe, A. Winfield

Intelligent Autonomous Systems Engineering Laboratory
Faculty of Engineering, University of the West of England
Coldharbour Lane, Frenchay, Bristol BS16 1QY, United Kingdom
email {ag_pipe, a-winfie}@uwe.ac.uk

Abstract

We describe an approach to automatic generation of local-cued reactive behavioral rules from the common features contained in a number of world models in the form of geometric spatial maps. It represents one stage of an ongoing effort to combine the best features of these two techniques in a complementary way. First we compare world modelling (or map-building) and reactive behavior as techniques for mobile robot controller design, identifying some of their major strengths and weaknesses, and briefly reviewing some examples of similar work in the context of our overall objectives. We then cover the following aspects of our overall architecture; the nature of the maps from which maze trajectories are derived, gathering sensorimotor data from an established trajectory, fuzzy clustering of the data to form rules, two variants of a reactive fuzzy-controller, and some experiments conducted in simulation to establish the viability of the approach. To date we are optimistic about the potential for these techniques in enabling an agent to build reactive behaviors autonomously. However there is much more work to be done, some of which we identify at the end of this paper.

1 Introduction

Combining some form of global mapping function with a localised reactive system in a single cohesive control architecture is clearly a hard but potentially powerful approach to the mobile robotics problem of navigation in continuous valued unknown, uncertain environments that have a dynamic element. Researchers from the communities of adaptive behavior, cybernetics and robotics have made a wide range of suggestions about how to devise a favourable mix [eg. Connell 1992, Kurz 1996, Mataric 1992]. The work presented in this paper forms part of *our* overall strategy to design such an architecture capable of learning to complete non-trivial orienteering tasks efficiently by attempting to utilise the best features of global spatial mapping and behavior-based robotics. However successful completion of this task requires a number of stages. In this paper we describe the stage which we have just completed and try to set it in the context of our overall approach.

In working towards the overall objective we became interested in assessing the feasibility of an agent capable of generating useful new reactive behaviors autonomously via a new two-stage learning paradigm. The results of these efforts form the main thrust of this paper. Our approach is to have the agent first learn a number of favourable trajectories through mazes via our established map-building mode of operation [Pipe et al 1, 2 & 3 1994, Pipe et al 1995] and then abstract from them any local sensory scenes they share so as to link them with appropriate actions, thus forming stimulus-response pairs. Whereas the maps that our agent builds contain *declarative* spatial information referenced to a global cartesian frame, the abstracted behaviors use only egocentric local sensory input, encoding *procedural* sequences. The focus of this paper therefore is on an autonomous process of transforming information about particular mazes so as to build and use new behaviors of a general nature, ie. a process of *induction*. It is perhaps appropriate to view the map-building mode as a "teacher" providing example situations from which may be extracted useful rules for behavior.

We chose the Fuzzy Logic [Zadeh 1965] paradigm as a natural embodiment of our reactive controllers. It handles continuous input-output values whilst being less internally "opaque" than a neural network. We were very interested in seeing what rules the system would abstract for itself.

In the remainder of this paper we first briefly set the scene for our overall approach by reviewing some critical characteristics of map-building and reactive agents plus some examples of existing work which uses them both in a single agent. We describe our simulated robot and its environment then briefly review those aspects of the internal representation and trajectories generated by the map-building agent which are necessary to understand the thrust of this paper. We go on to describe the induction process, a basic implementation of the reactive mode's internal architecture

(ie. the fuzzy controller), and an enhancement of it. We then present initial results from simulation and finish with conclusions and suggestions for further work.

2 Map-Building and Reactive Agents

For the purposes of this paper we find it useful to divide single-agent mobile robotics research into two categories representing each end of a spectrum of work. On one hand there are those who focus on the agent's possession of some structured, typically quite complex, form of internal representation of the world. This is normally an explicit component of the agent, the content of which is commonly built up autonomously through an exploratory process of discovery for which we use the term *map-building* in this paper. On the other hand there are those who feel that much useful work may be achieved without any explicit internal representation, indeed some have argued that none at all is required [eg. Brooks 1991]. We use the term *reactive* behavior to denote these "memoryless" stimulus-response components. Recent representative examples of the above two approaches may be found in [SAB90, SAB92 & SAB94].

However if we consider these two extremes then it seems clear that neither on their own accurately reflect the commonplace situation in nature. Most "higher" individuals in the animal kingdom are able to draw upon a huge range of behavioral modes as the situation demands, from the simplest reactive mechanisms to those which are hard to explain by any other means than the building of some powerful internal representation of the world. There is even evidence to suggest that elements of this large range could extend to the (so called) "lower" animals, especially in the case of some foraging insects [eg. Gallistel 1990].

For map-building mobile agents disambiguation of one local sensory scene from a similar one in another location is usually inherently simple, thus allowing different action sequences to be triggered for each if required. However this is clearly a non-trivial problem for many reactive robots, especially those which adhere strictly to the "no internal representation" tenet. Other planning tasks such as cul-de-sac avoidance, internalised short cut discovery, and planning routes between new goals are also commonly possible for map-building agents but hard or impossible for their reactive counterparts to perform.

However the building and maintaining of such internal representations is problematic. Firstly it tends to be computationally intense which can easily become critical in real-time situations, especially where the environment contains some dynamic component. By contrast reactive mobile robots have typically much lower processing overheads and therefore the speed of reactive ability is usually high. Secondly it seems difficult to attain the correct compromise between information content and accuracy requirements, especially when using acquired spatial information directly over extended distances or in complex environments. In particular there is usually a need for long term integration of sensory data in order to maintain position on an internal map.

We have certainly found that, whilst powerful in the ways mentioned above, using our map-building agent directly in the perception-action loop incurs a considerable processing overhead and is not suitable for operating in dynamic environments. It would seem that a more direct procedural form of knowledge might be better suited to the on-line determination of action on a short timescale, mediated on a longer timescale by interpretation of a map to resolve situations which appear ambiguous when only local sensory information is used.

There has certainly been much successful activity in the area of combining map-building and reactive control, we briefly describe some salient points of three researchers' approaches. Mataric [Mataric 1992] proposes the direct incorporation of an essentially topological map into the framework of a behavior-based Subsumption architecture [Brooks 1986]. This approach lends itself well to handling dynamic elements in an environment. However a certain amount of low resolution geometric information is still required in order to disambiguate sets of local sensory information which represent similar but different locations from those which actually do represent the same place. A similar approach is proposed by Connell [Connell 1992]. Kurz [Kurz 1996] adopts a more directly integrated geometric and topological approach to mapping, using a variety of techniques to ensure the usefulness of his more precise requirements for global geometric information. He does not use the Subsumption architecture as an underlying structure, however local sensory data is used along with the global information to identify "situations" on a directed graph. Kurz also reports favourable results of his approach in environments with a small dynamic element, using a "forgetting" algorithm to restructure the graph as the environment changes. The general proposal we make here for future work in completing our overall objective is to take a similar approach to Mataric and Connell in adopting a Subsumption-like architecture for low level reactive control. However we intend to use our directly geometric mapping approach [Pipe et al 1, 2 & 3 1994, Pipe et al 1995] to mediate between behaviors.

However if we shift the focus of discussion back to the more detailed subject of this paper, we may ask the following questions.

Why should we use a learning approach to acquire

behaviors at all? First of all we are simply interested in pursuing autonomy as far as possible, a reason in its own right for using an unsupervised learning technique. For our purposes this rules out hand-coding, which is in any case usually a far from trivial task and inevitably limits the agent to the stimulus-response set originally envisaged by the designer.

Why use the approach proposed here? Using some form of evolutionary or life-time reinforcement learning directly at the level of candidate behaviors/rules is an alternative which has been applied successfully to the case of "crisp" rules [eg. Wilson 1987, Roberts 1991, Pipe & Carse 1994], and "fuzzy" rules [eg. Parodi & Bonelli 1993, Grefenstette 1992, Berenji & Khedkar 1992]. Evolutionary reinforcement at the level of whole neuro-controllers [Cliff et al 1993, Husbands et al 1996] has also met with a measure of success. However when the agent is to operate in a non-trivial continuous valued environment then these tasks also become very difficult. By contrast although a map-building mode of operation is typically far from simple in itself if we assume that its use is desirable for other reasons, as we have argued here, then such an agent is already capable of generating a preferred trajectory around any number of static obstacles of arbitrary shape. Once this has been done the processes of periodically linking local sensory input with a given trajectory around an object (or objects) to generate a data set, and subsequent production of local-cued behaviors proves to be relatively straightforward.

3 The Simulated Robot and Environment

The robot is based upon a real one built in our laboratory which is illustrated in figure 1. It is set on a circular base approximately 150mm in diameter. It has two driving wheels mounted in a conventional manner so that they are able to provide both propulsion and steering. The set of sensors which the simulated robot possesses are;

a) a shaft encoder on each wheel,
b) five ultrasonic distance sensors set at the following angles from the "straight ahead" position; 0°, 90° to the left, 45° to the left, 45° to the right, and 90° to the right, each with a 2 meter maximum sensing range,
c) a "bump stop" which skirts the robot's perimeter.

When moving in free space under the command of a reactive fuzzy controller item a) is used by a low level controller which merges a constant forward velocity component of 0.1m/s with a simple inverse kinematics algorithm transforming body centered "steering angle" requests into the required number of extra shaft encoder pulses to achieve the turn. If collision with an obstacle is detected via item c) the forward component is reduced to zero yielding a "turn on the spot" effect due to steering angle commands. Item b) is used as a distance sensing array to provide input to the fuzzy controller. A system clock provides "time stamp" pulses every 100ms.

The maze is set on a rectangle of any size. Any number of rectangular obstacles of any shape or size may be placed in the maze. The start and goal positions likewise may be placed anywhere. In the experiments described here we used three generalised "building block" shapes from which to build mazes; 90° turns to the left, 90° turns to the right, and "crossroads". We should stress that choosing rectangular shapes for the obstacles and the maze itself was purely an expedient in generating the maze simulation. The agent itself has no such restrictions in its sensory or motor parts, in fact all measurements made, movements executed and internal spatial representations developed by the robot are continuous real valued, ie. there is no concept of a "grid" or discretized state space.

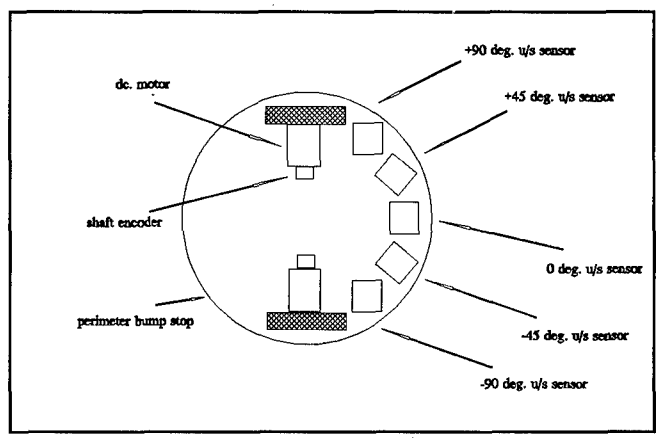


Figure 1

4 Spatial Maps Generated by Our Agent

This paper builds on our previous published work in map-building agents. A much longer paper than this would be required to describe the internal operation of the agent. However this has been done extensively elsewhere, including progressive developments and a range of examples of its application [Pipe et al 1, 2 & 3 1994], comparison with an alternative form of direct rule building architecture based around a Classifier System [Pipe & Carse 1994], and the effects of sensorimotor noise [Pipe et al 1995]. In many respects the work presented here begins where that work ended, ie. with trajectories representing converged solutions to particular mazes. In fact any agent capable of building some form of internal representation powerful enough to solve deceptive mazes of the type described above, and illustrated later in this paper, could be used to generate the optimal trajectories around obstacles which are required here. We wish only to give the reader a feeling for the

nature of the maps built by *our* agent which is sufficient to develop the more general points of this paper as identified in sections 1 and 2.

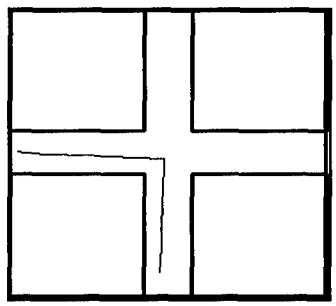


Figure 2

Each map built by the agent is stored in a 2-input/1-output Radial Basis Function (RBF) neural network of conventional form [Poggio & Girosi 1989, Sanner & Slotine 1991]. The two inputs are the (x,y) coordinates of a position in a given maze referenced to a global cartesian frame, and the output is an estimate of "value" for that position in terms of the expected travelling distance to some goal identified *a priori*. The inherent local generalisation of RBF networks allows a continuous representation to be developed from a search which is inevitably discrete and bounded.

For illustrative purposes a simple 2m X 2m maze is shown in figure 2, along with a converged trajectory, where the start is at the left and the goal at the bottom of the figure. For the RBF network a regular 32 X 32 array of gaussian basis functions of the following form was used.

$$act = \exp\frac{-dist^2}{(spread \times rbfdist)^2}$$ Where;

act = neuron activation,
dist = Input vector euclidean dist. to basis function center,
rbfdist = euclidean distance between basis function centers,
spread = constant affecting degree of local generalisation.

All other details of the map-building agent were as defined in [Pipe et al 3 1994].

Figure 3 attempts to show the contents of the map graphically. There are a number of factors to bear in mind whilst inspecting this figure. We found that features were more visible if we turned the z-axis upside down, ie. "good" places appear as valleys of positive fitness in the landscape, whilst "bad" places appear as ridges of negative fitness. Zero fitness is approximately 3/4 of the way up the z-axis shown on the left side of the figure. The agent cannot look into obstacles so only their edges are represented with meaningful values, the internal spaces simply reflect the RBF network weight vector initialization.

The start position is on the back-left and the goal is at the front-left. In addition places which the agent did not investigate fully will not be well represented in the map.

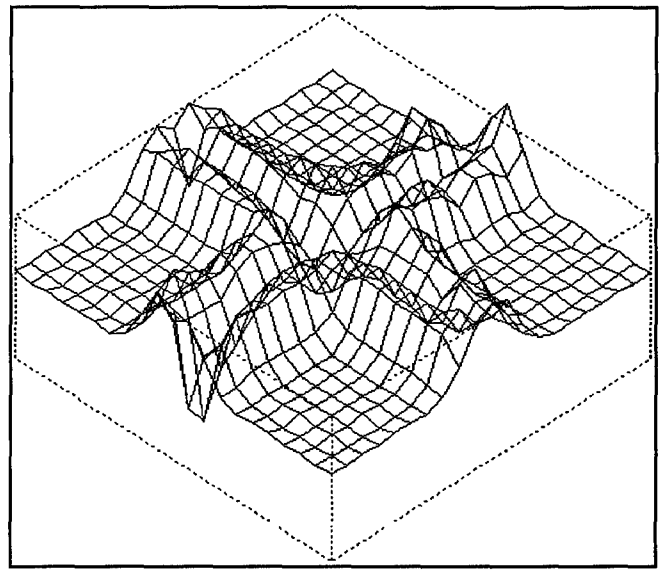


Figure 3

Turning this map into a trajectory is simply a matter of searching from any start point for the best allowable straight line move in terms of maximising the fitness increase. "Allowable" in this sense means that the trajectory does not pass through an area of negative fitness on the map, ie. a known obstacle. Whilst the map is being learnt of course this internalised scheme does not always work because it will, to one extent or another, be incomplete. However this is an issue of further map-learning which is beyond the scope of this paper [see Pipe et al 1 1994]. However if we assume that the map is sufficiently complete then finishing the trajectory is a matter of repeating this process until the goal is reached as shown in figure 2.

An alternative view of such a map is that it contains a probability density function which approximates the environment in terms of the likelihood of encountering obstacles, sub-goals and the overall goal based on evidence from the search to date. In this context one could view it as a "fuzzy" map in the sense that the Radial Basis Functions play a very similar role to input membership functions in the fuzzy logic paradigm [see Jang & Sun 1993], it certainly does not convey "crisp" information. In experiments conducted elsewhere [Pipe et al 1995] we have shown that this kind of spatial map remains useful in the presence of quite high levels of sensorimotor noise and bias.

5 Rule Induction

There are two main phases, gathering sensorimotor information, and fuzzy clustering.

Phase 1 is as follows. The robot builds a map. It then uses the information contained in it to move from a start position to a goal, thus generating a trajectory around

obstacles in the maze. As it moves through the maze along this trajectory it takes distance measures to nearby obstacles using all five ultrasonic sensors at 100ms intervals, recording these along with a body-centerd steering angle relative to the current heading. Each one of these measurements thus forms a 5-input/1-output relation.

This phase will typically be repeated one or more times for the same, or different, mazes. Since this is a real valued environment even if it is done on the same maze no two routes will be identical and this helps to improve robustness in the resulting behavioral rule set. There is an element of "hands-off" supervision here. Clearly the original choices of maze shape and start/goal positions will determine the content of the sensorimotor information to be clustered and ultimately the fuzzy rule set of a reactive controller. This is reminiscent of the term *scaffolding* [Rutkowska 1994] sometimes used in developmental psychology to describe the way in which parents structure the learning experiences of their infants. In the next section we describe two approaches to structuring the learning experience each of which is used in conjunction with a different controller architecture.

The second phase of rule induction uses the data acquired by either of these means in an off-line process of rule extraction, or fuzzy clustering. There are several well established clustering algorithms for use with fuzzy logic, the best known group being the fuzzy c-means algorithm [Bezdek 1981] and its variants. In this technique a set of c fuzzy membership functions have their centers moved until they best represent partitions in the N data items, where the number of membership functions is set beforehand at $2 \leq c < N$. However for this work our requirements are not complex and we adopt a simpler method identified by Wang & Mendel [Wang & Mendel 1992] which simply uses a set of fixed center membership functions. The algorithm consists of five steps.

Step 1 Divide the input and output spaces into fuzzy regions. After some experimentation we chose 5 membership functions on each input and 17 on the output, a fuller description is given in the following section.

Step 2 Generate a fuzzy rule from each input-output relation in the given data set. This is simply a process of working through each input-output relation in the data set linking the maximally active membership function to each input and output value. When step 2 is finished we have as many rules as there are input-output relations in the data set. A large number of them will inevitably not represent particularly good matches with membership function centers. A great many of them are likely to be linked with the same input membership functions but have conflicting output linkages.

Step 3 To resolve these problems a strength measure for each rule is derived from the product of each input's membership function activation (in the conventional sense of activation) for a given input.

Step 4 The conflicts described in step 2 are resolved using the strength values from step 3, thus generating the rule base.

Step 5 Use this rule base as the fuzzy controller with "live" sensory input.

6 Reactive Mode - The Fuzzy Controller

For our first experiments we chose a fairly basic form of Fuzzy Logic control system. When active as the robot's controller it is run through one forward pass every 100ms system clock cycle, simply providing an updated steering angle for that period.

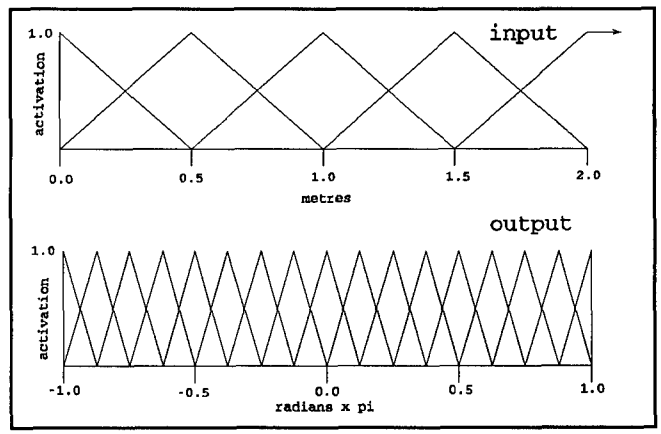


Figure 4

We chose a conventional distribution of unit-height triangular membership functions with all functions being identical and equally spaced across the domain of discourse, with the exception of each input function that was placed at the maximum value of an input variable. Here we assumed that the ultrasonic transducer has a maximum range of 2m. We therefore placed an asymmetric function with a ramp at the left side peaking at 2m but continuing on the right side at a height of 1 unit effectively infinitely, thus providing an activation of 1 for any distance greater than 2m. Figure 4 illustrates this distribution of membership functions for one of the inputs and the output.

For fuzzy AND we used the product of membership function activations for a given rule as opposed to the simpler MIN operator since it requires little extra processing and is known to produce superior interpolation properties [eg. Harris 1992]. Defuzzification was performed by conventional center of gravity calculations.

Using the basic approach described so far we devised an alternative scheme for constructing a fuzzy logic based reactive controller. In the original approach we simply use all derived rules together in a single system, ie. the conventional fuzzy logic paradigm, which is suitable for those situations where a single maze contains the complete range of behaviors to be abstracted. The alternative approach is more suited to an incremental process of behavior abstraction where a number of mazes contain one or two disparate behavioral situations each.

We constructed a "winner takes all" bidding system in which a number of (typically smaller) fuzzy logic controllers compete for control of the agent at each 100ms system clock cycle. We borrowed this structure from a generalised competitive behavior-based approach to robotics [eg. Brooks 1986] to which it clearly bears some resemblance.

Competition takes place on the basis of comparing the degree of matching between each fuzzy logic system and the current sensory input. Since such a measure is available as a natural by-product of fuzzy inference, ie. fuzzy AND, this incurs little processing overhead. In this paper we use the terms *single* and *composite* in reference to these two fuzzy controller architectures.

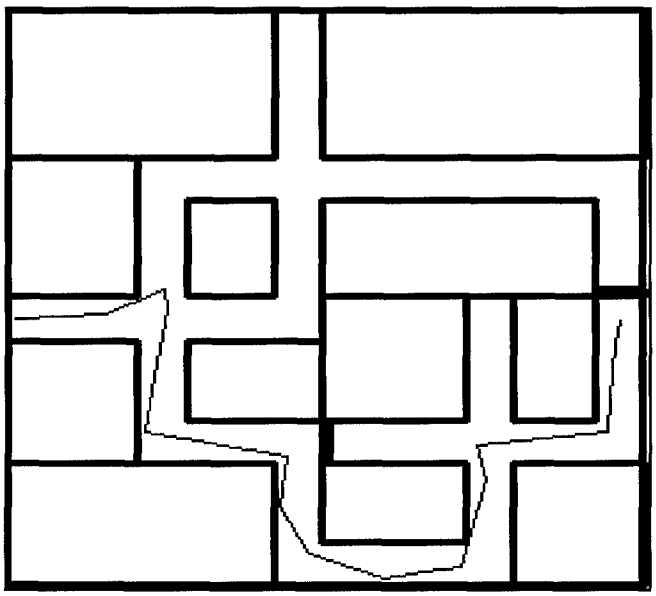


Figure 5

7 A Single Fuzzy Controller Experiment

In this experiment a single fairly complex maze was solved three times in map-building mode to yield a set of similar maps, and thereby trajectories. Figure 5 shows the maze, set on a 4m X 4m square with the start position on the left and the goal at the right. The trajectory generated by the map-building agent during one of the sensorimotor "data gathering" runs is also shown in figure 5.

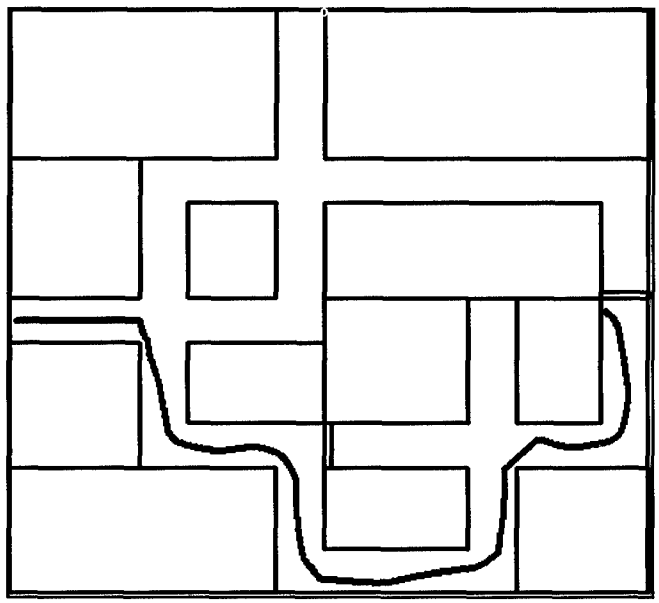


Figure 6

After rule induction the fuzzy controller based agent was, as one would hope, capable of negotiating a similar route through this maze as shown in figure 6, though the trajectories are somewhat different in detail. This is largely because the map-building agent always moves in straight lines followed by stationary turning phases in order to reduce sensory processing overhead, whereas the fuzzy-controller combines steering and forward motion as a virtually continuous process.

It is perhaps also worth noting that since the reactive fuzzy-controller is operating without intervention of the map-building operational mode there is no concept of a goal. This simulation was terminated at the point shown on the right in figure 6, but if it was continued the agent would simply turn around and move back through the maze, turning right into each dead end of the crossroads junction and so on indefinitely.

For this example 59 rules were generated. Four examples are shown below for illustrative purposes. We were quite pleased by the relatively simple process of relating the presence of particular features with rules for a behavior. For example it was quite easy to spot the rules responsible for "turn right at crossroads", "winding corridor following" and "turning round at dead end" (we sometimes started our map-building agent pointing straight at a dead end rather than out into the maze so that this became part of the trajectory).

These rules are listed below.

input					*output*	*description*
0	*45l*	*90l*	*45r*	*90r*		
1.0	0.5	1.0	0.5	1.0	-0.5	cross_right
0.0	0.5	1.0	0.0	0.0	+0.5	corridor_left
0.5	0.0	0.0	0.5	1.0	-0.5	corridor_right
0.0	0.0	0.0	0.0	0.0	-1.0	deadend_turn

Where the following abbreviations and conventions apply.

The left five columns denote the ultrasonic sensor inputs, eg. *45l* = input from sensor pointing 45° to the left of center. Numbers in each column represent the center of a membership function for that input expressed in meters.

The numbers in the *output* column are the membership function centers for each rule expressed in radians x π about a positive-anticlockwise z-axis.

Above we have illustrated four of the most "interesting" rules, but most rules had a steering angle output close to zero, along with what *appeared* to be a few critical rules for turning corners. However, thinning out or removing the zero output rules by hand soon revealed that most were equally as important in maintaining a straight-line trajectory when this was required. Without them, weakly firing "turning" rules would predominate under inappropriate circumstances making the agent unstable.

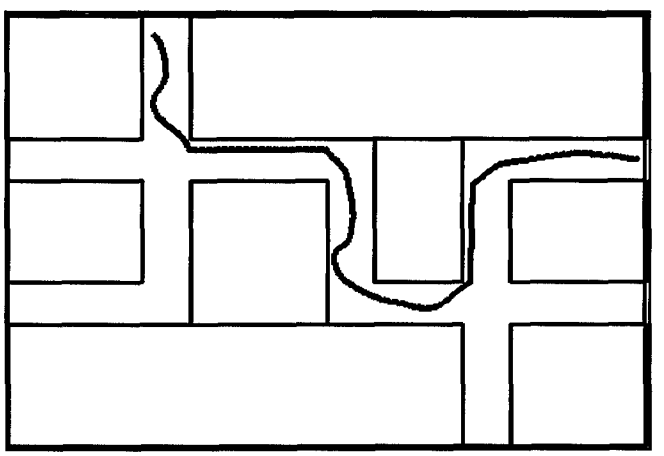


Figure 7

Next we tested the fuzzy controller on a new 3m X 4m maze, ie. one for which no process of map-building and rule extraction had been performed. We made no special effort to maintain exact dimensions of corridors and junctions and in fact the maze shown in figure 7 has dimensions which are different from those found in the original maze by a random amount up to a maximum of +/-15%. Figure 7 also shows the trajectory followed by the agent from right to left.

8 A Composite Fuzzy Controller

In this experiment we constructed two simple fuzzy rule bases, using the same basic procedure as before, from the mazes shown in figure 8 and figure 9. They were both set on 2m squares with the start position to the left on figure 6 and at the bottom in figure 9. In each case the figure also shows one of three map-building trajectories used to gather sensory data.

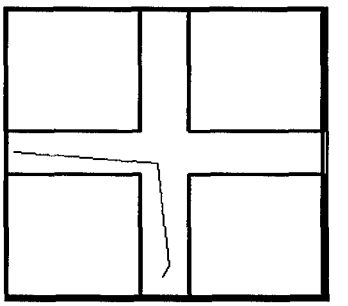


Figure 8

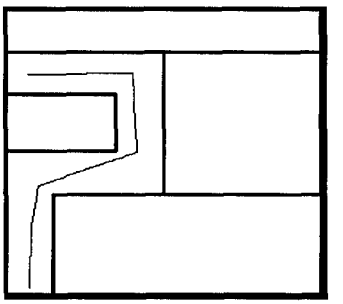


Figure 9

In this example 21 rules were generated for the "crossroads" maze and 34 for the "winding corridor" maze. Again as one would hope, this 2-component composite controller was capable of negotiating either of these two mazes. Next we tested it on the more complex maze of figure 10, which also shows the trajectory followed by the agent from right to left of a 2m X 4m maze. It was also able to negotiate the mazes shown in figures 6 and 7 successfully.

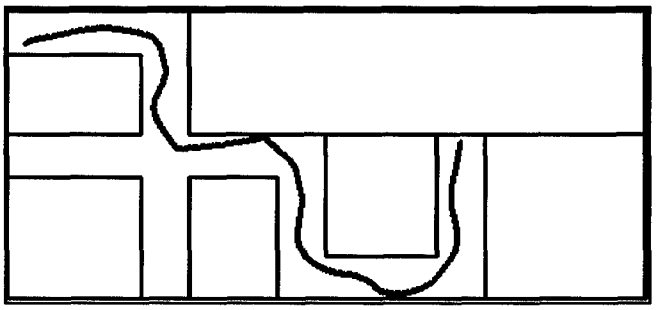


Figure 10

9 Adding Further Competencies

We were interested in establishing whether we could add new behaviors to an existing composite fuzzy controller in an incremental fashion without disrupting performance.

We constructed a very simple maze for the agent to solve in map-building mode, which merely consisted of a straight corridor with the start position at one end, the goal at the other, plus a small obstacle. This maze was then

solved three times by the map-building agent with the obstacle in different places. From this compiled trajectory data an additional fuzzy logic system consisting of 20 rules was extracted and then added as a third component to the existing composite controller from the previous section.

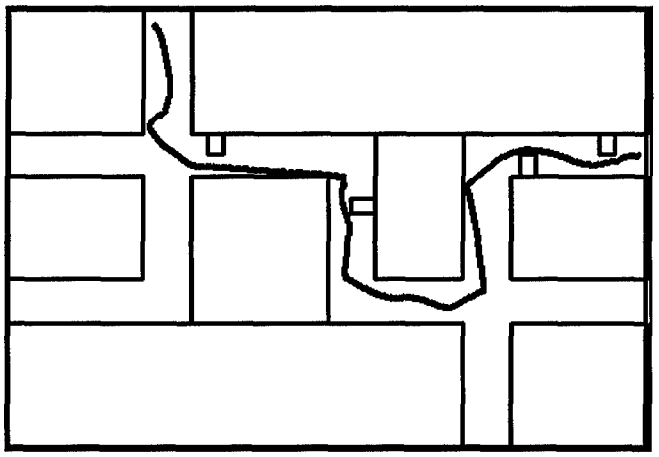


Figure 11

We first checked that the new controller could still negotiate the mazes shown in figures 7 and 10 satisfactorily. Having established this we then devised a 3m X 4m maze, shown in figure 11, which was the same as that depicted in figure 7 but with four of the small obstacles placed in locations where they would impede the natural trajectory of the original 2-component fuzzy controller from right to left of the figure. When we tested the original controller it did indeed get stuck in a continuous ricochet between the first two additional obstacles. However the 3-component controller successfully proceeded through the maze as shown.

10 Emergent Behavior

We tested the 2-component composite controller from section 8 on a number of new junction shapes which were not present in the mazes used for abstracting behaviors. One example of a T-junction is shown in figure 12 where the agent was started at the top. Here the "straight ahead" rules predominate until the junction approaches, which appears sufficiently similar to a combination of crossroad and right-bend for those rules to gain control of the robot. In fact the agent produced a very similar trajectory if it was run in the reverse direction. Generally it displays a propensity for taking any turnings which become available and is quite robust against collisions with obstacles. Of course this may not be the desired behavior pattern in order to solve some particular maze. Under these circumstances the robot must be returned to the map-building mode in order to generate a desirable trajectory from which appropriate new behavioral rules can be extracted. It would be possible either to increase the rule-base of an existing behavior or to add a new one, at present a human design decision dependent on whether the single or composite fuzzy-controller approach was to be adopted.

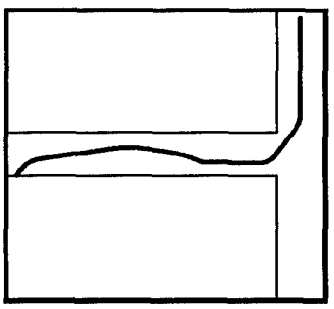


Figure 12

11 Robustness Tests

We have undertaken many experiments in attempting to establish the robustness of these fuzzy controllers to changes in the dimensions of mazes compared with the ones which were used to derive the rules. One example presented here was to test the ability of the 2-component composite controller derived in section 8 to execute a "turn right at crossroads" manoeuvre as the corridor widths were made wider or narrower relative to those used in deriving the fuzzy rules.

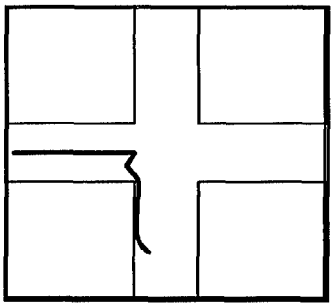


Figure 13

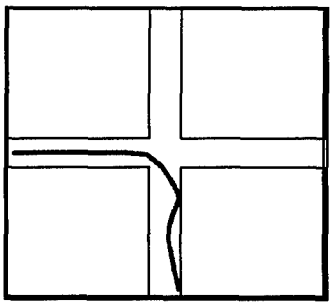


Figure 14

Figures 13 and 14 show the extreme cases where the agent just managed to produce a useful trajectory. Approximate limits were established at +50% and -50% respectively by comparison to the crossroads maze shown in figure 8. Further changes produced "circling in the center of the junction" behavior for wider corridors and "straight ahead" for narrower ones. As in the previous section clearly if the limits of a given behavior are exceeded then unuseful actions are likely to be the result and the only recourse is a return to the map-building mode to extract new behavioral rules.

12 Discussion and Conclusions

It would be a relatively simple task to hand craft the rule sets for fuzzy controllers with superior performance to the ones which our learning architecture extracted. However the emphasis here is on whether or not it is *possible* to complete

the path of autonomous learning from building a number of spatial maps of mazes, to trajectories derived from them, and then finally to generalised local-cued mobile robot behavioral rules. In this respect we find these results encouraging. There is a great deal of work in the literature concerning individual topics covered in this paper, some examples of which we have briefly considered. Of particular interest are those approaches which combine global mapping with local sensory information, however the authors are unaware of an overall approach which combines the techniques used here in the same way.

Clearly we have not finished the process of combining map-building and reactive operating modes. As mentioned previously the advantages of each might best be exploited by the following approach. A reactive controller would be used to dictate moves on a short time scale, coping with localised dynamic disturbances. At the same time approximate position on a global map would be maintained and used to override reactive behavior when necessary in order to disambiguate apparently conflicting local scenes. In fact although in our preliminary experiments we did not observe a great performance difference between the single and composite fuzzy control architectures, this wider objective provides an important reason for adopting the latter approach.

Let us assume that we have constructed a maze in which it is necessary to turn right at a first crossroad junction but left at the next in order to successfully solve it in the map-building mode. If we attempted to abstract rules for a single fuzzy controller then steps 3 and 4 of the clustering algorithm will choose one subset of rules for turning right, or the other for turning left, or worse a mixture of the two. This will either eliminate one of the desired behaviors or produce an unsatisfactory mix. However if we were to adopt a stepwise *scaffolding* approach to the acquisition of behaviors then we could execute the following procedure. We establish a first fuzzy rule base by solving a maze with a single "turn left" crossroads and then a second from a "turn right" maze. We then combine these two viable, but conflicting, fuzzy logic components into a composite controller along with other useful components. When solving the original two-crossroads maze discussed here the map-building system monitors progress in a "tracking mode". When two or more fuzzy components compete for control of the agent with high strength values then it intervenes to resolve the conflict.

When considering problems of non-trivial complexity proliferation of rules (or neurons in neural network approaches) is a problem which often has to be treated carefully in unsupervised learning systems, especially those employing evolutionary reinforcement. However we have found that use of a fuzzy clustering technique to generate rules has an inherent generalizing effect which is easy to control so that their number can be reduced to an acceptable level (typically less than 100 in total for all our experiments to date).

Although we have speculated about favourable performance of a reactive mode of control when confronted with dynamic environments we have not yet tested our agent in this way. Clearly this is urgent further work.

Many detailed enhancements are also clearly possible. Below are a few examples.

We chose a regular fuzzy membership function distribution throughout, fixed *a priori*. A more complex clustering algorithm might be devised in which determination of these factors became part of the rule extraction process. For example intuition led us to believe that in the output space the narrow closely packed membership functions we chose were really only required close to the origin.

There appeared to be some redundancy in the rule base with respect to some inputs. Certain groups could probably be reduced to a single rule which uses only some of the inputs, especially for those concerned with maintaining straight line movement. An automated parser which checks for this should be relatively simple to devise and could make the resultant rule base more robust.

A certain amount of on-line "tuning" of the rule base would in all likelihood be beneficial. This could take the form of a supervised learning algorithm acting on the centers of the output membership functions to minimise some criterion such as deviation from the center line of corridors.

Our preliminary results lead us to be optimistic about the potential for this approach, however much more testing is required and although the results presented in this paper were performed on a real-time graphical simulator, nothing substitutes for the real world and we will soon be testing this agent on the real robot in our laboratory.

References

Berenji H. R., Khedkar P., 1992, 'Learning and tuning fuzzy logic controllers through reinforcements', IEEE Trans. on Neural Networks, 3(5).

Bezdek J., 1981, 'Pattern recognition with fuzzy objective function algorithms', Plenum, New York.

Brooks R., 1986, 'A Robust Layered Control System for a Mobile Robot', IEEE J. Robotics and Auto., RA-2, no. 1.

Brooks R., 1991, 'Intelligence without Representation', Journal of Artificial Intelligence, Vol. 47, pp.139-159.

Cliff D., Harvey I., Husbands P., 1993, 'Explorations in Evolutionary Robotics', Journal of Adaptive Behavior, 2(1), pp.73-110.

Connell J. H., 1992, 'SSS: A hybrid architecture applied to robot navigation', Procs. of IEEE Int. Conf. on Robotics and Auto., Nice, France, pp.2719-2724.

Gallistel, C. R., 1990, 'The Organization of Learning', MIT Press.

Grefenstette J. J., 1992, 'The Evolution of Strategies for Multiagent Environments', Journal of Adaptive Behavior, vol. 1, no. 1, pp.65-90.

Harris C. J., 1992, 'Comparative aspects of neural networks and fuzzy logic for real time control', in Neural Networks for Control and Systems, IEE Control Eng. Series #46, chap. 5, Peter Peregrinus, pp.72-93.

Husbands P., Harvey I., Cliff., Thompson A., Jakobi N., 1996, 'The artificial evolution of control systems', Procs. of 2nd Int. Conf. on Adaptive Computing in Engineering Design and Control, Plymouth, pp.41-49.

Jang J. S. R., Sun T. S., 1993, 'Functional Equivalence between Radial Basis Function Networks and Fuzzy Inference Systems', IEEE Transactions on Neural Networks, vol. 4(1) pp.156-159.

Kurz A., 1996, 'Constructing maps for mobile robot navigation based on ultrasonic range data', IEEE Trans. on Systems, Man & Cybernetics, part B, vol. 26, no. 2, pp.233-242.

Mataric, M. J., 1992, 'Integration of Representation into Goal-Driven Behavior-Based Robots', IEEE Transactions on Robotics and Automation, Vol. 8, No. 3, pp.304-312.

Parodi A., Bonelli P., 1993, 'A new approach to fuzzy classifier systems', Procs. of 5th Int. Conf. on Genetic Algorithms, pp.231-237.

Pipe A. G. 1, Fogarty T. C., Winfield A., 1994, 'A Hybrid Architecture for Learning in Continuous Environmental Models in Maze Problems', From Animals to Animats 3, Ed. Cliff, Husbands, Meyer & Wilson, MIT Press, pp.198-205.

Pipe A. G. 2, Fogarty T. C., Winfield A., 1994, 'Hybrid Adaptive Heuristic Critic Architectures for Learning in Mazes with Continuous Search Spaces', Parallel Problem Solving from Nature (PPSNIII), Proceedings of the third International Conference on Evolutionary Computation, Springer-Verlag Lecture Notes in Computer Science #866, pp.482-491

Pipe A. G. 3, Jin Y., Winfield A., 1994, 'A Hybrid Adaptive Heuristic Critic Architecture for Learning in Large Static Search Spaces', Proceedings of the 9th IEEE International Symposium on Intelligent Control, IEEE Control Systems Society 94CH3453-8, pp.237-242.

Pipe A. G., Carse B., 1994, 'A Comparison between Two Architectures for Searching and Learning in Maze Problems', Springer-Verlag Lecture Notes in Computer Science #865, pp.238-249.

Pipe A. G., Carse B., Fogarty T. C., Winfield A., 1995, 'Learning Subjective "Cognitive Maps" in the Presence of Sensory-Motor Errors', Procs. of Third European Conference on Artificial Life, Lecture Notes in Artificial Intelligence 929, Eds. Moran, F., Morano, A., Merelo, J. J., Chacon, P., Springer, pp.463-476.

Poggio T., Girosi F., 1989, 'A theory of Networks for Approximation and Learning', MIT AI lab. Memo 1140.

Roberts G., 1991, 'Classifier Systems for Situated Autonomous Learning', PhD thesis, Edinburgh University.

Rutkowska, J. C., 1994, 'Emergent Functionality in Human Infants', From Animals to Animats 3, Proceedings of third International Conference on Simulation of Adaptive Behavior, MIT Press, pp.179-188.

SAB90, 1990, 'From Animals to Animats', Procs. of 1st International Conference on Simulation of Adaptive Behavior, Eds. Meyer & Wilson, MIT Press.

SAB92, 1992, 'From Animals to Animats 2', Procs. of 2nd International Conference on Simulation of Adaptive Behavior, Eds. Meyer, Roitblat & Wilson, MIT Press.

SAB94, 1994, 'From Animals to Animats 3', Procs. of 3rd Int. Conference on Simulation of Adaptive Behavior, Eds. Cliff, Husbands, Meyer & Wilson, MIT Press.

Sanner R. M., Slotine J. E., 1991, 'Gaussian Networks for Direct Adaptive Control', Nonlinear Systems Laboratory, MIT, Cambridge, USA, Technical Report NSL-910503.

Wang L. X., Mendel J. M., 1992, 'Generating fuzzy rules by learning from examples', IEEE Transactions on Systems, Man and Cybernetics, vol. 22, pp.1414-1427.

Wilson S. W., 1987, 'Classifier Systems and the Animat Problem', Machine Learning 2(3), pp.199-228.

Zadeh L., 1965, 'Fuzzy Sets', Journal of Information and Control, vol. 8, pp.338-353.

MOTIVATION AND EMOTIONS

A New Control Architecture Combining Reactivity, Planning, Deliberation and Motivation for Situated Autonomous Agent

François Michaud, Gérard Lachiver and Chon Tam Le Dinh
Department of Electrical and Computer Engineering
Université de Sherbrooke
Sherbrooke (Québec) CANADA J1K 2R1
{michaudf, lachiver, ledinh}@gel.usherb.ca

Abstract

Intelligent behavior can be observed from both natural and artificial systems, but still the notion of intelligence is very difficult to define. We propose a new control architecture allowing the combination of reactivity, planning, deliberation and motives for building intelligent agents that must deal with real or other kinds of environments suited to their life purposes. This control architecture tries to unify the principles and characteristics associated with intelligence. It uses behaviors as its basic control components. These behaviors are selected dynamically and their actions are combined according to the intentions of the agent. Introspection of its reactions and its knowledge is one major new ability given to the agent by this architecture. This way, the agent is not only able to adapt to the environment, but also to its own capacities. One implementation for experimenting with a simulated environment for mobile robots is presented here to illustrate the use of this architecture.

1. Introduction

Intelligence studied in its various aspects is a source of inspiration in Artificial Intelligence (AI). Insights about intelligence are taken from psychology [Simo95], from neuro-ethology [Beer90], from personal intuition and from observation of our own intelligent behavior. In spite of the significant progress made in AI, intelligence is still a difficult notion to define and to entirely reproduce in artificial systems. The fields of research in AI follow particular guidelines about the principles associated with intelligence. For example, Saridis [Sari83] believes that intelligence is based on a three-level hierarchy (execution, coordination and organization), layered according to the decreasing precision with increasing intelligence principle. Albus [Albu91] proposes also a hierarchical architecture but with a central world model and according to a functional decomposition of intelligence. In contrast, multi-agent research [Wern92] believes in distributing intelligence in different knowledge sources that have to work together to solve a task. The behavioral decomposition of intelligence proposed by Brooks [Broo86] also proves to be a very useful approach. Finally, hybrid approaches try to combine the advantages of reactivity and planning by following the previous guidelines [Firb89, Nore95, Simm90, Haye95, Kael86, Donn94].

All of these principles are associated with intelligence, and a way to combine them into a general architecture is needed. A more general architecture must allow the combination of characteristics associated with intelligence like reactivity, emergent functionality, modeling, planning, deliberation, learning, goal, motivation and emotion. But these characteristics are not sufficient nor essential conditions to characterize or to reproduce intelligent behavior into an artificial system. Intelligent behavior must be established based on the ability of the agent to adapt to two things: the environment it is in; and its own capacities or limitations (of its sensing and actuating abilities, its processing and memorizing abilities, and its decision abilities) affecting its ability to interact with the environment.

This paper presents a new control architecture that tries to take into consideration all of these aspects and to combine them while preserving their underlying principles [Mich96]. Section 2 presents the architecture and its characteristics. Section 3 describes the use of this architecture and the mechanisms developed for experimentation using a simulated world for mobile robots. Section 4 shows some results obtained with the simulated environment to illustrate the use of the architecture. Finally, conclusions and future work are outlined.

2. Intentional Selection of Behaviors

This new architecture presents interesting characteristics for building intelligent agents that must deal with real or other kinds of environments suited to their life purposes. It consists of six modules as illustrated in Figure 1. The *Behavioral* module is a behavior-based system [Broo86, Mata92]. It is made of different behaviors connecting sensory information to actuation. It defines the agent's skills for reacting to the situations encountered in the environment. These behaviors all run in parallel and their resulting commands are blended according to their respective importance, to obtain the control actions. The relevance in using each behavior is determined by three recommendation modules. The *External Situation* module evaluates special external conditions in the environment that can affect behavior selection. The *Needs* module selects behaviors according to the needs and goals of the agent. The third recommendation module, called the *Cognition* module, is for cognitive recommendation. This module learns things about

the external environment and how the agent operates in it by observing its reactions, behavior selections and from information sensed from the environment. It can then exploit this acquired knowledge or some innate knowledge to plan or to prepare the use of behaviors. Cognitive recommendations can be influenced by behaviors via the *Internal Parameters* link, like they can influence behavior reactivity using the same link. These three recommendation modules suggest the use of different behaviors to the *Final Selection* module, which combines them appropriately to establish the activation of the behaviors (*Behavior Activation*). An active behavior is allowed to participate to the control of the agent, and it is said to be exploited if it is used to control the agent (by reacting to the sensations associated with the purpose of the behavior). Finally, the *Motives* module is composed of motives used to examine and to coordinate the proper working of the other modules. Motives are influenced by the environment, the internal drives of the agent, its knowledge, and by observing the effective use of the behaviors (*Behavior Exploitation*). The agent is then able to adapt its emerging behavior to its needs, its knowledge and its ability to satisfy its intentions.

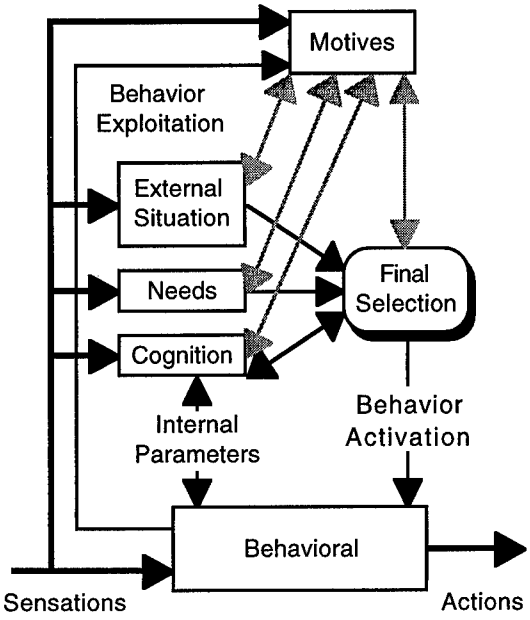


Figure 1: Control architecture proposed

This is a very brief explanation of the modules. However, a better description can be made using an example. The next section describes the mechanisms implemented for the experiments using a simulated world for mobile robots.

3. Module Implementation for the Simulated World

To validate the control architecture proposed, we wanted to develop an autonomous agent having to deal with various goals like managing its energy, its purposes and its well being. To do this, we have used a simulated world for mobile robots called *BugWorld* [Alm93]. An agent in *BugWorld* has a circular body equipped with distance sensors (similar to range finders) for detecting obstacles and targets. For our experiments, the agent is placed in a room where it can find targets and a charging station. The agent must be able to efficiently reach the targets and must survive by recharging itself when needed. The agent knows nothing about the environment it is in, but it has a limited memory to acquire knowledge that can be helpful in its task. For sensing, the agent has at its disposal eight proximity sensors for obstacle, each separated by 45° starting from its nose. There are also two target sensors, one on each side. One target in the room is used as a charging station. The agent can also read the amount of energy available, its speed and its rotation (the rotation is only used to indicate when the agent is moving or not). For actions, the behaviors can affect the speed, the rotation or a variable for the color of the agent.

To achieve this task, the agent is going to follow a scenario guiding its general behavior according to what it experiences in the environment. First, the agent starts to acquire knowledge about the environment by following boundaries, reaching a target or a charging station deliberately or not. When the agent is able to recognize its location in the environment, it can start exploring other regions. Eventually, when the agent judges that it knows enough about the environment, it can use this knowledge for reaching memorized targets.

Different AI methodologies can be used with the modules of our control architecture. For the experimentation presented here, the implementation of the modules are inspired by three techniques from other approaches. Fuzzy logic is used for behaviors and for the blending of their control actions, like in Saffiotti *et al.* [Saff93]. It is also used by the *External Situation* module and the *Needs* module for recommending behaviors, and by the *Final Selection* module for combining these recommendations. A topological graph, having some similarities with the work of Mataric [Mata92], is used by the *Cognition* module to construct an internal representation of the environment based on the experiences of the agent. Finally, activation levels as in Maes [Maes91] are used for motives. All of these techniques are described in the next subsections.

3.1. *Fuzzy Behaviors*

A fuzzy behavior uses rules and linguistic variables to establish the relation between sensations and actions. The processing steps are similar to the ones for fuzzy systems [Lee90], which are fuzzification, rule inference and defuzzification. The only difference here is that rule firing strength is affected by μ_{act}, the activation of the behaviors, given by the *Final Selection* module. The processing steps for the *Fuzzy Behavioral* module are summarized below:

- Fuzzification (1). This operation converts input data into linguistic values (A_i) characterized by a label and a membership value. Figure 2 gives an example of membership functions used for fuzzification of the front

sensor of the agent (i.e. its nose). In this case, the fuzzy representation of a sensory input of 15 is given by two linguistic values: a membership of 0.67 for the *Danger-in-front* fuzzy set, and a membership of 0.33 for the *Near-front* fuzzy set.

$$Sensations \rightarrow \mu_{A_i}(Sensation) \quad (1)$$

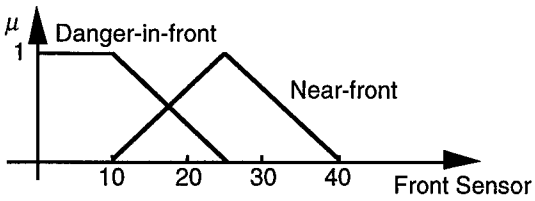


Figure 2: Example of membership functions

- Fuzzy implication of rule r for the behavior j (2). The operator ⊗ is the minimum and is used for the fuzzy conjunction of the n antecedents of rule r. The result is called the firing strength of the rule, which is a measure of the contribution of the rule to the fuzzy control action [Lee90]. The firing strength of the rule is associated with its fuzzy consequence, which is a linguistic variable (B or C) for a fuzzy control action. This processing step is repeated for all the rules of a behavior, and all the activated behaviors. When NOT is used in front of an antecedent, the complement [Lee90] of its membership value is used. Examples of rules are presented in Figure 3 for the EMERGENCY behavior, and in Figure 4 for the TURN180 behavior. For the rule *Slow-down-danger* of Figure 3, if *Danger-in-front* has a membership value of 0.67 and *Speed-null* has a membership value of 0 (which indicates that the agent is moving), then the membership value of *Slow-down-fast* is 0.67.

$$\mu_{B_{rj}}(Action) = \otimes\left[\mu_{A_n}(Sensation)\right] \quad (2)$$

- Adjustment of the firing strength of the rules according to the activation μ_{act} of the behavior j (3). The minimum is also the fuzzy conjunction operator used here. For example, if the activation of the EMERGENCY behavior is 0.5 and the firing strength of rule *Slow-down-danger* is 0.67, then the adjusted membership value of the fuzzy consequence *Slow-down-fast* for this rule is 0.5.

$$\mu_{C_{rj}}(Action) = \otimes\left[\mu_{B_{rj}}(Action), \mu_{act}(j)\right] \quad (3)$$

- Union of the fuzzy consequences (4). The fuzzy disjunction operator ⊕ maximum is used to combine membership values for identical fuzzy consequence. For instance, the rule *Danger-in-front* of the EMERGENCY behavior and the rule *Immobilization* of the TURN180 behavior have the same fuzzy consequence. If their respective adjusted membership values for *Slow-down-fast* are 0.5 and 0.9, then their union gives a membership value of 0.9.

$$\mu_{C_o}(Action) = \oplus\left[\mu_{C_{rj}}(Action)\right] \quad (4)$$

- Defuzzification using the center of area method (5). This step converts the fuzzy consequences into a 'crisp' (numerically precise) output. The parameter w_{Cx} is the support value at which the membership function for C_x reaches the maximum value μ_{Cx} (the average is used when there is two support values with the same membership strength), and x represents the linguistic variables for a common control action. This step allows the smooth blending of the fuzzy commands given by the activated and exploited behaviors. For example, the *Slow-down-fast* fuzzy consequence is a linguistic variable for the *Acceleration* control action (associated with speed control). Its w parameter is -3.25 for a membership value of 0.5. For the same control action, if there is another fuzzy consequence called *Accelerate* with a membership value of 0.1 and w = 4, then the resulting control action *Acceleration* is -2.04.

$$Action = \frac{\sum_x \mu_{C_x}(Action) \cdot w_{C_x}}{\sum_x \mu_{C_x}(Action)} \quad (5)$$

Twelve behaviors are used in our experiments. The first is EMERGENCY, which is responsible for moving the agent when immediate danger is detected in its front. The rules are presented in Figure 3. Using this behavior, the agent slows down if it is in front and very close to an obstacle; it turns away from an obstacle at its side (the variable x is for *left* or *right*, and y denotes the opposite direction); and it makes a wide turn left when the obstacle is right in its front.

```
<Slow-down-danger>
    IF      Danger-in-front
    AND     NOT (Speed-Null)
    THEN    Slow-down-fast
<Danger-x>
    IF      Danger-front-x
    AND     NOT (Danger-front-y)
    THEN    Turn-y
<Danger-in-front>
    IF      Speed-Null
    AND     Danger-in-front
    AND     Danger-front-right
    AND     Danger-front-left
    THEN    Turn-left-big
```

Figure 3: Rules for the EMERGENCY behavior

Other behaviors are AVOID to move away from obstacles, SPEED to maintain a constant cruising velocity, ALIGN to follow boundaries, TARGET to search for a target, RECHARGE to search for a charging station and to energize the agent, BACKING to move back, MADNESS to make the agent turn around on itself, TURN90 to move away from a boundary, TURN180 to make a U-turn, ALARM to express some internal state of the agent by changing its color to red, and a behavior for identification of topological states (used by the *Cognition* module: see Section 3.4). Rules for the TURN180 behavior are presented in Figure 4. The behavior starts by slowing down the agent. It then makes it turn away from a boundary until the other side is perceived by the agent. The underlined antecedents and consequences of the rules are adjusted by the *Cognition* module via the *Internal Parameters* link before using the behavior, to select the rotation side.

```
<Immobilization>
    IF      NOT(Speed-almost-null)
    THEN    Slow-down-fast
<Turn-180-left>
    IF      NOT(Side-left-near)
    AND     Speed-almost-null
    THEN    Turn-left
<Turn-180-left-end>
    IF      Side-left-near
    AND     Back-near
    AND     Speed-almost-null
    THEN    Turn-left
```

Figure 4: Rules for the TURN180 behavior

3.2. *Motives*

The agent has five basic goals in the environment: to find a charging station and recharge itself; to reach targets; to detect improper use of its behavior according to its intentions; to explore the environment and acquire knowledge from it; and to use this knowledge when the agent judges it is accurate enough. Motives are responsible for coordinating and supervising these goals according to the actual experiences of the agent in the environment. To do so, ten motives are used by the agent. Each one has its own activation mechanism and can be influenced by sensations (internal or external), behavior activation or exploitation, internal variables of the recommendation modules, or by other motives. Four groups of motives are used:

- Physiological motives, like HUNGRY and EAT, are used to monitor the energy level of the agent and to control the use of the RECHARGE behavior via the *Needs* recommendation module. The agent shows opportunism by wanting to recharge when it reaches a charging station, even if it is not hungry;

- Good Operation motives. These motives are particularly influenced by the *Behavior Exploitation* link to detect improper use of behaviors. Because behaviors are fuzzy, *Behavior Exploitation* is a fuzzy measure defined in relation (6), approximating the contribution or the importance of behavior j to the fuzzy control actions formulated before defuzzification. It combines the activation of a behavior with its reactivity to the environment.

$$\mu_{exp}(j) = \mu_{act}(j) \otimes \left(\oplus \left[\mu_{B_{rj}}(Action) \right] \right) \qquad (6)$$

Two motives are in this group. The motive DISTRESS is used to monitor the proper working of behaviors like EMERGENCY, AVOID and SPEED. These first two behaviors must normally be exploited very briefly to move the agent away from trouble areas. However, if their μ_{exp} remains approximately constant for a long period of time, this may be a sign of conflict between the behaviors used. For the SPEED behavior, a full exploitation for a long period of time is also a sign of trouble indicating that the agent is not able to reach its desired velocity. The motive DECEPTION is the other motive in this group. This motive increases when the agent is moving away from a target or a charging station, detected by a decrease in the exploitation of the TARGET or RECHARGE behaviors respectively. This motive influences the use of the TURN180 behavior via the *Cognition* module;

- Accomplishment motives. Only one motive, called FULFILLMENT, is used in this group to monitor when the agent needs to find targets;

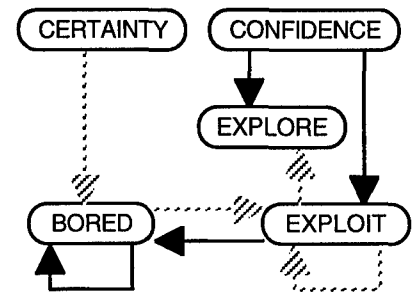


Figure 5: Cognition motives

- Cognition motives supervise the acquisition and the use of the knowledge in the *Cognition* module. These motives are illustrated in Figure 5. A solid arrow indicates a positive influence from another motive, as opposed to a shaded one. The motive CONFIDENCE is associated with the agent's ability to locate itself in previously memorized topological sites. When the agent is able of doing so, the motives EXPLORE and EXPLOIT are excited. The motive EXPLORE directly uses the level of excitation of CONFIDENCE to

influence the cognitive recommendation of the TURN90 behavior. The motive EXPLOIT increases gradually when CONFIDENCE is greater than zero, indicating the increasing ability of the agent to know where it is in the environment. EXPLOIT is also excited when the topological graph is full. When EXPLOIT is sufficiently excited, it inhibits the motive EXPLORE to stop the acquisition of knowledge and to use the topological graph constructed for going toward memorized targets. The ability to plan a path using the topological graph is reflected by the CERTAINTY motive. During the exploitation of the topological graph, the BORED motive increases when no path is planned (toward some target that has not already been visited during the exploitation of the topological graph) or when no target is reached. Eventually, the motive BORED is fully excited and reinitializes the EXPLOIT motive. Exploration is then resumed.

3.3. *External Situation and Needs*

These two modules recommend the use or the inhibition of behaviors. They are implemented using fuzzy logic. The operations are similar to those presented in relations (2) and (4), except that the results are fuzzy measures of the desirability or the undesirability of behaviors. The difference between these two modules is that the *Needs* module can use motives as antecedents in its rules. Figure 6 shows the rules used by the *External Situation* module, and Figure 7 presents rules for the *Needs* module. In these rules, an undesired behavior is a consequence preceded by NOT.

<Danger>
IF Danger-in-front
OR Danger-front-right
OR Danger-front-left
THEN EMERGENCY
<Obstacle>
IF Obstacle-in-front
THEN AVOID AND NOT(TARGET)
<Normal>
IF NOT(Obstacle-in-front)
THEN SPEED AND ALIGN
<Topological states>
IF NOT(Speed-null)
AND NOT(Rotation-null)
THEN TOPOLOGICAL STATE IDENT.
<Charging>
IF Speed-almost-null
AND Charging-station-visible-left
AND Charging-station-visible-right
THEN NOT(ALIGN)

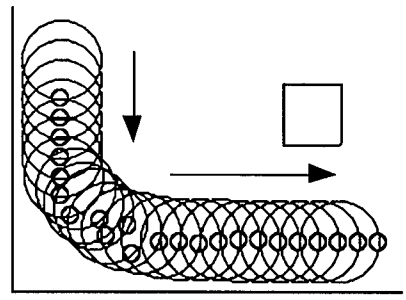
Figure 6: Rules for the *External Situation* module

<Want-to-recharge>
IF Want-Recharge
THEN RECHARGE, NOT(MADNESS)
NOT(TURN90) AND NOT(TARGET)
<Charging-station-near-*x*>
IF Want-Recharge
AND Charging-station-visible-*x*
THEN NOT(SPEED)
<Charging-station-nearer-*x*>
IF Want-Recharge
AND Charging-station-nearer-*x*
THEN NOT(ALIGN)
<Difficulties>
IF Distress-exists
THEN BACKING, ALARM
AND NOT(ALIGN)
<Accomplishment>
IF Fulfillment-small
THEN TARGET
<Happiness>
IF Fulfillment-big
THEN MADNESS, NOT(SPEED)
AND NOT(ALIGN)

Figure 7: Rules for the *Needs* module

3.4. *Topological Graph*

Knowledge, acquired by the agent from its experiences in the environment, is represented using a topological graph. The graph is constructed from topological states identified by the TOPOLOGICAL STATE IDENTIFICATION behavior. This behavior examines sensations coming from the front, the back and the two sides of the agent. The presence or the absence of obstacles at a certain distance in these four directions is used to infer one of 16 possible topological states. Figure 8 gives an example of topological states identified as the agent turns a corner and passes near an obstacle while following boundaries.



Right side - Right side - Right side - Right side - Right side - Right corner to turn - Right corner to turn - Dead-end right - Dead-end right - Right corner turned - Right corner to turn - Right corner turned - Right corner turned - Right side - Right side - Right side - Right side - Right side - Right side - Right side - Corridor - Corridor -Corridor

Figure 8: Example of identification of topological states

Two types of nodes are constructed using these topological states according to their characteristics: identification of the same topological state like *Right side*, *Left side*, *Corridor* or *Nothing*, for a consecutive number of cycles, is associated with a stable landmark node; other topological states identified between two landmark nodes are used to construct a transition node. For a transition node, the topological state sequence is analyzed to characterize and to approximate the rotation made by the agent. This is done using regular expressions for a lexical analysis [Aho88] of the topological states identified during this sequence. Figure 9 presents some of the regular expressions used. On the right side of the expressions, the operators indicate the number of the same consecutive topological state that must be in the sequence: '+' indicates more than one; '-' indicates only one; '?' is for zero or one. These expressions are evaluated in parallel after the consecutive identification of the same topological state. If a topological state does not correspond with the active state in a regular expression, then the regular expression is dismissed. Priority is assigned according to the order of definition of the regular topological expressions (the first ones have priority on the others). When an expression is completely validated, the result at the left side of the expression is memorized. If there is no result obtained for the sequence of topological states identified during a transition, the best guess, according to the expressions evaluated, is used. These regular expressions can be seen as symbolic behaviors reacting to topological state sequences to characterize the transition made by the agent.

Internal *Right* Corner - 110° =
Right corner to turn+ Dead-end *right*+
Right corner turned- *Right* corner turned-
Right corner turned-
Internal *Right* Corner - 90° =
Right corner to turn+ Dead-end *right*+
Right corner turned- *Right* corner turned?
External *Right* Corner - 110° =
Nothing+ *Right* side+ Nothing+
External *Right* Corner - 90° =
Nothing+ *Right* side - Nothing -
U Turn *Right* - 180° =
Right corner turned- Against a wall-
Left corner turned-

Figure 9: Example of regular topological expressions

Nodes are constructed as the agent moves in the environment. They memorize the landmark type (the topological state for a stable landmark, or the result obtained from a transition landmark analysis), its length, the orientation of the agent (for stable landmark only, cumulated from previous nodes according to the rotation approximated by transition nodes) and the number of the branch in construction in the graph. Other information like the number of visits to the node, the presence of a goal like a target or a charging station, the occurrence of a motive like DISTRESS or DECEPTION, the use of particular behaviors like TARGET and MADNESS, and path planning variables are other fields accessible in a node. Nodes are connected together with bidirectional links. These links have information about the anticipation of the state of the connected node, making the graph reversible. Uncertainty measure concerning the length of the node, which occurs between a landmark node and a transition node, is also memorized in the link. Figure 10 presents the graph constructed based on the topological states identified in Figure 8.

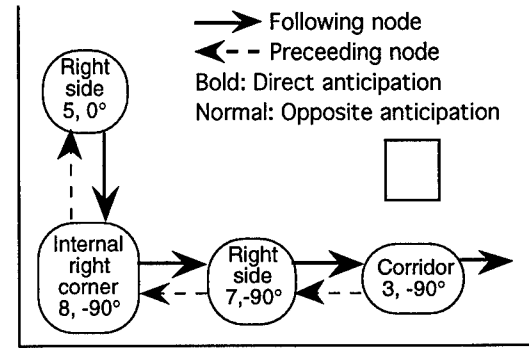


Figure 10: Topological graph

To find out if the agent is located on a previously visited landmark, a search in the constructed nodes must be initiated. During this search, the *Cognition* module tries to find a sequence of three nodes similar to the three most recent nodes constructed. Similarities are evaluated according to the landmark type, the anticipation of the link followed to establish the sequence, the length of the nodes, and sometimes the orientation. When only one similar sequence is detected, the recent nodes can be eliminated and a loop can be established in the topological graph. The agent is then situated in its memorized topological representation and can compare its state according to the following nodes in the graph. If a divergence is observed, a new branch is initiated and the search process is reactivated. In addition, when the graph is full or when the agent wants to exploit it, three buffer nodes are used to locate, if possible, the agent in its topological graph.

Path planning with the topological graph is done by activation spreading from the current location to a node referring to a particular goal. Activation is spread following the links to the nodes, preferring the paths in the same direction of the agent, with the fewest nodes, the smallest length, and avoiding special conditions like DISTRESS, DECEPTION, MADNESS and U-turns. Path planning is also used to optimize the topological graph when the agent wants to exploit it. Only the useful nodes are then kept in the graph. A node is considered to be useful when it has been visited more than once, when it indicates the start of an exploration path, or when it is part of the optimal path from a node referring to a charging station to a node referring to a target.

Cognitive recommendations are binary (which is a special case of fuzzy membership strengths of 0 or 1). They are responsible for making the agent explore the environment by using the TURN90 behavior, and also for making a U-turn by using the TURN180 behavior. The SPEED and ALIGN behaviors are then inhibited to let these behaviors control the actions of the agent. Cognitive recommendations are also responsible for the use of special behaviors when a memorized topological path must be reproduced. Finally, they can also activate the ALARM behavior to transmit an S.O.S. when the agent thinks it does not have enough energy to reach a charging station (based on what it can anticipate from its topological graph).

3.5. *Final Selection*

The fact of using desirability and undesirability for recommending behaviors has been inspired by the hedonic axiom which indicates that the organisms direct their behaviors to minimize aversions and maximize desirable outcomes [Beck83]. Here, *Behavior Activation* is evaluated based on a hedonic continuum established from the fuzzy desirability and undesirability measures. First, these measures are respectively combined for each behavior using the fuzzy disjunction operator maximum. Then, the desirability is subtracted to the undesirability measure and the behavior is activated if the result is greater than zero. Relation (7) shows these operations where *m* represents the recommendations from the three recommendation modules. So, to be activated, the desirability of the behavior must be higher than its undesirability.

$$\mu_{act}(j) = \max\left(0, \oplus\left[\mu_{des_m}(j)\right] - \oplus\left[\mu_{und_m}(j)\right]\right) \quad (7)$$

4. Experimental Results

For our experimentation, the agent has enough energy for 250 cycles. When it reaches a charging station, the agent detects that it is recharging by sensing an increase of its energy level. It stops recharging when its energy level is maximum. Also, a target reached is inhibited for 200 cycles. The results presented here are parts of longer trajectories followed by the agent. We only want to show the use of the mechanisms described in Section 3 and to illustrate the general behavior of the agent in two different environments.

Figure 11 illustrates the initial trajectory followed by the agent starting from a given point in the environment, and Figure 12 presents the activation level of some of its motives. This environment comes with *BugWorld*. The agent starts by reaching the upper left corner target and continues its path by following boundaries. It stops at the lower left corner where the charging station is located. It then continues to follow boundaries, reaching the lower right target and the upper right target successively. The FULFILLMENT motive is increased by 30% when a target is attained. After that, the agent reaches again the upper left target and the charging station. The EAT motive is fully activated when the agent is recharging, confirming that the behavior RECHARGE is used. At this point, the agent is able to detect similar sequences in its topological graph and to construct a loop in it. The agent has a higher level of CONFIDENCE and wants to EXPLORE. It starts exploring the center of the room just after leaving the charging station by using the TURN90 behavior at a point when the agent feels it is able to move away from the boundary without being influenced by other obstacles.

The agent continues to explore the environment this way, only using the TURN90 behavior at proper landmarks when it has never been used. After a while, the agent feels confident for long periods of time because it is able to locate itself in its topological graph and it cannot explore new sites in the environment. As illustrated in Figure 13, the motive

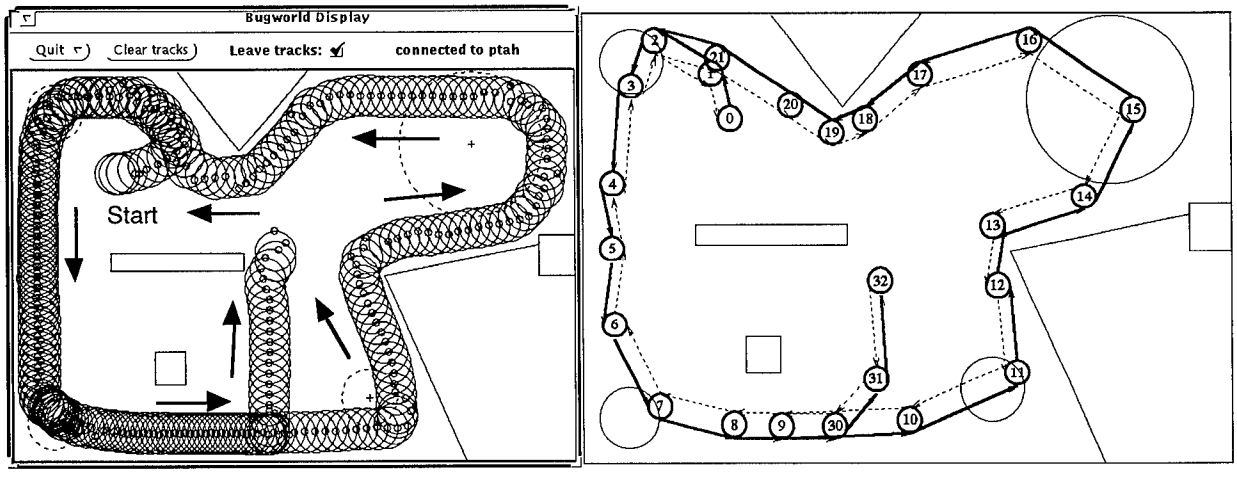


Figure 11: Trace and topological graph observed when the agent starts exploring the environment

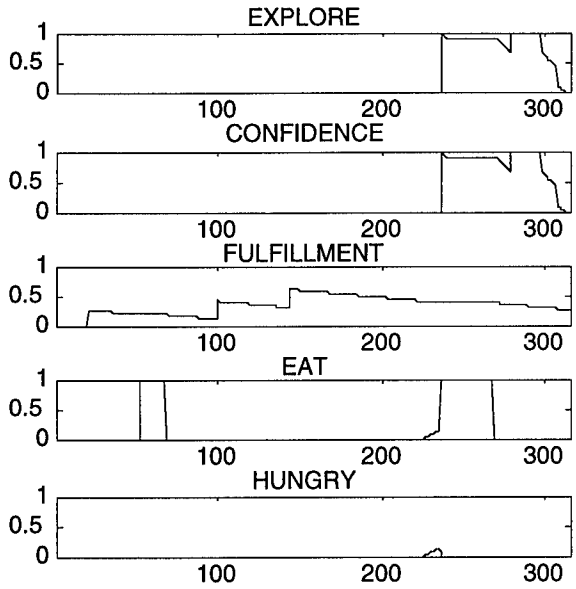


Figure 12: Important motives for the trace presented in Figure 11

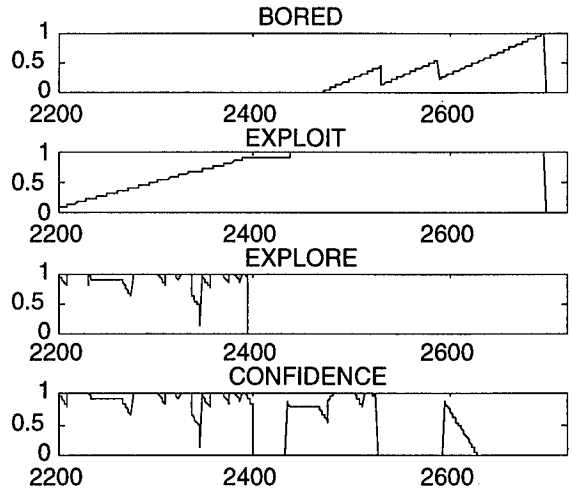
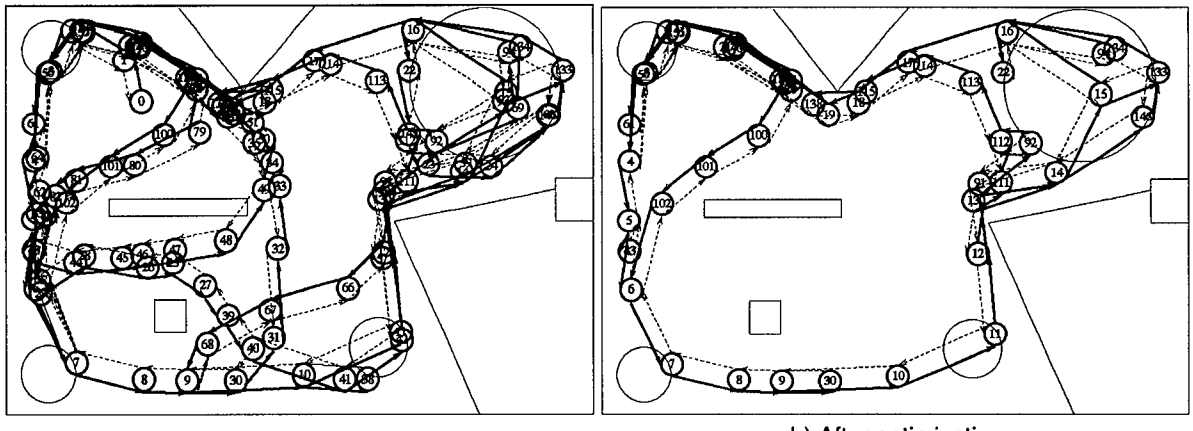


Figure 13: Motives when the topological graph is exploited

EXPLOIT increases until it reaches a preset value of 0.9. It then inhibits the EXPLORE motive, and the buffer nodes are used to locate the agent in its graph. The EXPLOIT motive reaches full activation when the agent arrives at a charging station and takes the time to optimize its topological graph. Then, the agent can use its graph to plan a path toward a target. In the case presented here, the agent is able to use its buffer nodes for positioning in the topological graph but at a certain point, the agent is unable to know where it is (as we can see by the zero CONFIDENCE level before and after the 2600 step). The agent is also unable to plan paths or to reach targets during that time, and the motive BORED increases until it resets the EXPLOIT motive.

Figure 14 shows the topological graph before and after optimization. As illustrated in a), because similarities between nodes are evaluated based on a sequence of similar nodes and not by factors uniquely identifying each possible landmark in the environment, the graph can be composed of parallel branches or multiple nodes for the same site. To resolve this possible confusion, we can see that it is important to optimize the graph. In b), the graph is reduced by approximately 50%. The paths following the boundaries of the environment are kept without being explicitly specified in the optimization procedure. This way, the useful paths emerge from the experiences of the agent in the environment and its abilities to use this representation.

In Figure 15, the agent starts from another point in the environment. Because of a conflict between EMERGENCY, AVOID and ALIGN behaviors, the agent gets stuck in the lower right corner. The simultaneous constant exploitation of EMERGENCY and AVOID excites the motive DISTRESS from which the BACKING behavior is recommended by the *Needs* module. The agent then starts moving towards the charging station, but observes a decrease in the exploitation of the TARGET behavior. This indicates that it is moving away from a target which, in this case, is the upper right target. The motive DECEPTION is then increased and the agent makes a U-turn by using the



a) Before optimization b) After optimization

Figure 14: Topological graph before and after optimization

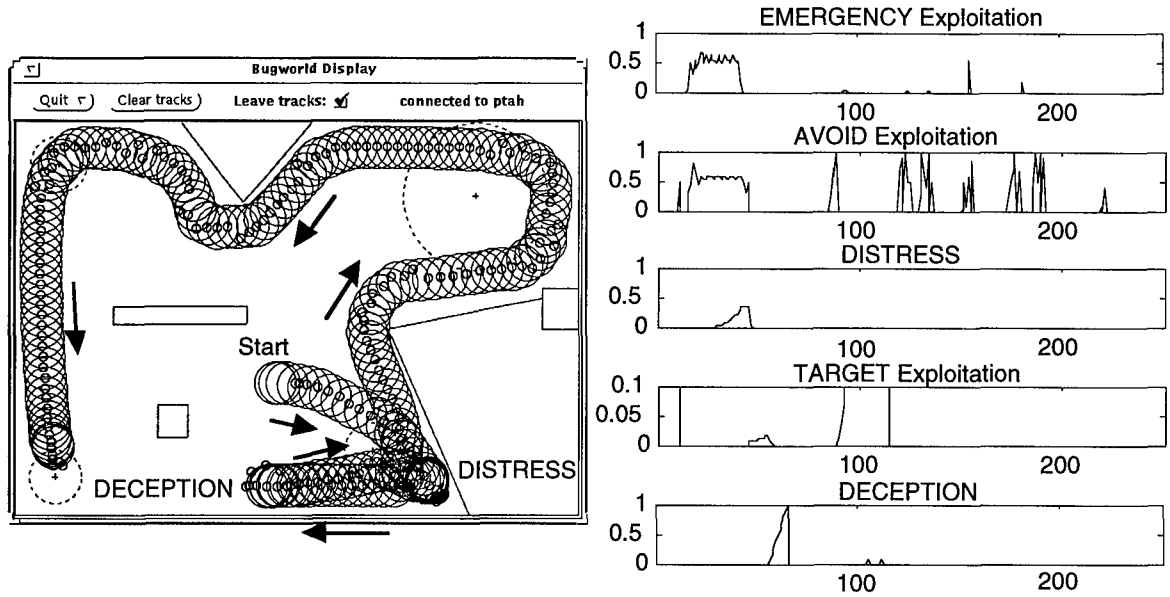


Figure 15: Trace and conditions for the motives DISTRESS and DECEPTION

TURN180 behavior. The agent continues its path by following boundaries until it reaches the charging station. An S.O.S. is emitted before arriving at this point because the agent has only three cycles of energy left.

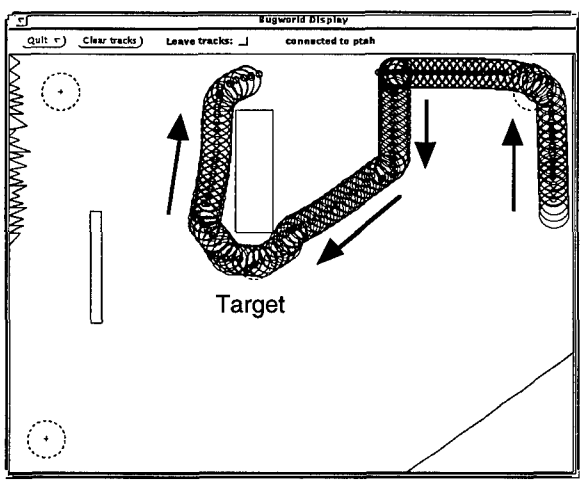


Figure 16: Path reproduction

Other environments have also been used during our experimentation. Figure 16 shows two parts of a trace made by the agent placed in another room. The first one is made when the agent explores the environment. The agent is then able to reach the target at the center of the room. Later, when the agent exploits its topological graph, it is able to plan a path toward this target and to reproduce the path using the buffer nodes and its optimized topological graph. Figure 17 shows a special condition which occurred when a moving obstacle was placed in the same room with the agent. At one point, the obstacle is moving toward the agent. The agent tries to move away from the obstacle, but cannot do so because its back side collided with the moving obstacle. The agent does not have any behaviors to avoid obstacles from its back. But, by observing that the SPEED behavior is fully exploited for a long period of time (because the agent wants to move but simply cannot get some speed), the motive DISTRESS is excited so that the BACKING behavior can be used.

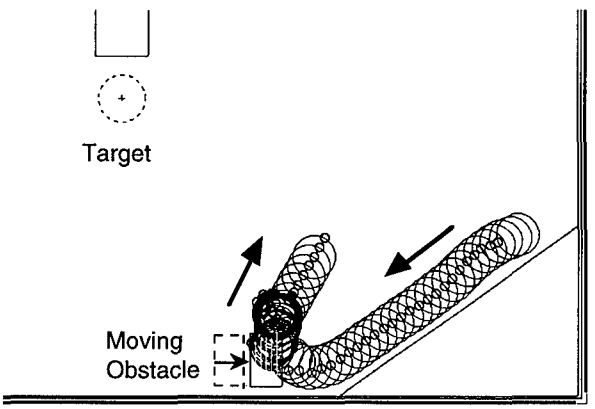


Figure 17: DISTRESS when a mobile obstacle came toward the agent

The emergence of functionality is very important in the mechanisms implemented for these experiments. Emergence is considered in behavior reactivity and parallelism, the fusion (or blending) of the control actions, behavior modification via the *Internal Parameters* link, and the dynamic selection of behaviors. Emergence is also considered for the fuzzy recommendation modules, the motives and the observation of *Behavior Exploitation*. Finally, the *Cognition* module and the topological graph considered emergence of the representation used by the agent.

This representation is constructed and managed directly from the actions and the internal capacities of the agent. Using this characteristic in all the architecture modules, the agent has showed its ability to adapt to the environment and to its own limitations of interacting with it.

5. Conclusion

This article is too brief to give all the details about the architecture proposed along with the description of the mechanisms used for the experimentation presented here, and references to related concepts from artificial or natural systems. The objective of the article is rather to introduce this architecture and to demonstrate the possibility and the usefulness of combining reactivity, planning, deliberation and motivation. This architecture is based on five hypothesis: intelligence is behavior-based; intelligence depends on the internal context and the external context; intelligence emerges holistically; introspection is a basic constituent of intelligence; and intelligence is affected by the autonomy of the system. All of these facts are important in making the agent adapt to its own reality. Using this architecture, an intelligent functionality emerges from the agent's interactions with its external environment and its internal intentional senses. Other types of mechanisms could be used in its modules and modules can be used, if needed, according to the purpose of the system to be controlled. This way, the architecture tries to unify the different views, principles, mechanisms and characteristics associated with intelligent behavior.

Acknowledgments

Support from the Natural Sciences and Engineering Research Council of Canada (NSERC) was highly appreciated. We would also like to thank Nikolaus Almàssy for letting us use *BugWorld.*

References

[Aho88] Aho, A.V., Sethi, R., and Ullman, J.D., *Compilers, Principles, Techniques, and Tools*, Addison Wesley, 1988.

[Albu91] Albus, J.S., "Outline for a theory of intelligence", *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 21, no. 3, pp. 473-509, May-June 1991.

[Alm93] Almàssy, N., "BugWorld: A distributed environment for the development of control architectures in multi-agent worlds", Tech. Report 93.32 (ftp anonymous, almassy@ifi.unizh.ch), Dept. Computer Science, University Zurich-Irchel, December 1993.

[Beck83] Beck, R.C., *Motivation. Theories and Principles*, Second Edition, Prentice Hall, 1983.

[Beer90] Beer, R.D., *Intelligence as Adaptive Behavior. An experiment in computational neuroethology*, Academic Press, 1990.

[Broo86] Brooks, R.A., "A robust layered control system for a mobile robot", *IEEE Journal of Robotics and Automation*, vol. RA-2, no. 1, pp. 14-23, March 1986.

[Donn94] Donnart, J.Y. and Meyer, J.A., "A hierarchical classifier system implementing a motivationally autonomous animat", in *Proc. Third Int'l Conf. on Simulation of Adaptive Behavior,* The MIT Press, 1994, pp. 144-153.

[Firb89] Firby, R.J., "Adaptive execution in complex dynamic worlds", Ph.D. Thesis, Dept. Computer Science, Yale University, 1989.

[Haye95] Hayes-Roth, B., "An architecture for adaptive intelligent systems", *Artificial Intelligence*, vol. 72, pp. 329-365, 1995.

[Kael86] Kaelbling, L.P., "An architecture for intelligent reactive systems", in *Proc. Workshop on Reasoning about Actions and Plans,* 1986, pp. 395-410.

[Lee90] Lee, C.C., "Fuzzy logic in control systems: fuzzy logic controller", *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 20, no. 2, pp. 404-435, March-April 1990.

[Maes91] Maes, P., "A bottom-up mechanism for behavior selection in an artificial creature", in *Proc. First Int'l Conf. on Simulation of Adaptive Behavior,* The MIT Press, 1991, pp. 238-246.

[Mata92] Mataric, M.J., "Integration of representation into goal-driven behavior-based robots", *IEEE Trans. on Robotics and Automation*, vol. 8, no. 3, pp. 304-312, 1992.

[Mich96] Michaud, F., "Nouvelle architecture unifiée de contrôle intelligent par sélection intentionnelle de comportements", Ph.D. Thesis, Dept. Electrical and Computer Engineering, Université de Sherbrooke, 1996 (in French).

[Nore95] Noreils, F.R. and Chatila, R.G., "Plan execution monitoring and control architecture for mobile robots", *IEEE Trans. Robotics and Automation*, vol. 11, no. 2, pp. 255-266, April 1995.

[Saff93] Saffiotti, A., Ruspini, E., and Konolige, K., "A fuzzy controller for Flakey, an autonomous mobile robot", Technical Note 529, SRI International, March 1993.

[Sari83] Saridis, G.N., "Intelligent robotic control", *IEEE Trans. on Automatic Control*, vol. AC-28, no. 5, pp. 547-557, May 1983.

[Simm90] Simmons, R., Lin, L.J., and Fedor, C., "Autonomous task control for mobile robots", in *Proc. Fifth Int'l Symposium on Intelligent Control,* 1990, pp. 663-668.

[Simo95] Simon, H.A., "Explaining the ineffable: AI on the topics of intuition, insight and inspiration", in *Proc. Int'l Conf. Artificial Intelligence (IJCAI),* Montréal, August 1995, pp. 939-948.

[Wern92] Werner, E., "The design of multi-agent systems", in *Proc. Third European Workshop on Modeling Autonomous Agents in a Multi-Agent World,* Elsevier Science, 1992, pp. 3-28.

A Finer-Grained Motivational Model of Behaviour Sequencing

Emmet Spier and David McFarland

Animal Robotics Laboratory

Animal Behaviour Research Group
Zoology Department, Oxford University
South Parks Road, Oxford OX1 3PS, UK.
emmet.spier@ox.ac.uk, mcfarland@vax.ox.ac.uk

Abstract

In this paper we consider three points. 1) Ethological behaviour models are not grounded in exhibited behaviour - the models have not been developed far enough, indeed, work in the animat community may help ethology; 2) In order to assess agent performance we should try to develop a functional account of behaviour whilst developing a mechanistic account; 3) Planning and reasoning is not necessary to exhibit sophisticated appetitive behaviour sequences. To illustrate these points, our model of behaviour selection in a self-sufficient autonomous agent is based on an ethological motivation model which extends traditional ethological modelling in the direction of implementable behaviours. The model is firstly justified from a functional approach and then tested and shown to have improved performance in a two-dimensional continuous simulated environment. It is noted that the model exhibits what could be described as planning behaviour.

1 Introduction

We are interested in the class of agents which are autonomous and self-sufficient; that being, agents that ensure that they will never run out of fuel or place themselves in a situation that will lead to their running out of fuel (McFarland and Bosser, 1993). Such agents must sequence their behaviour between multiple tasks, at the minimum, maintaining their fuel supply and, also, performing some other additional task to, perhaps, earn their keep (McFarland and Bosser, 1993; McFarland and Spier, 1996).

Sequencing of behaviour has received a great deal of interest and the problem has seen many approaches, for instance Agre and Chapman (1987), Rosenblatt and Payton (1989) and Maes (1990) in the animat literature as well as from the more classical approaches in artificial intelligence. We agree with Brooks (1990) that deliberative methods for action selection have not lived up to their promise and with Wilson (1991) in considering that methods of behaviour sequencing need to be constructed with regard to the niche within which the agent exists.

By accepting the constraint of self-sufficiency we gain several benefits, not least in providing a base-line from which we can judge the behaviour sequencing of an agent. The agent may have the constraint incorporated within for assessing its own behaviour and determining its priorities without the requirement of an externally provided goal state. In addition, the constraint of self-sufficiency has many parallels with the survival constraint within a biological context and, as such, it is tempting to draw upon the history of ideas from ethology (Halperin, 1991; McFarland and Bosser, 1993; Tyrrell, 1993b; Blumberg, 1994, Spier and McFarland, date).

Tinbergen (1951) suggested that ethologists ask four questions when they enquire about the behaviour of an animal: 1) What is the function of the behaviour? 2) What is the mechanism that exhibits the behaviour? 3) What is the ontogeny of the behaviour? and 4) What is the evolutionary history of the behavioural trait? The first two questions are particularly relevant not only to the study of animal behaviour but to the design and investigation of animats.

This paper considers the use of both functional and mechanistic methods of analysis in the design of a behaviour sequencing system for self-sufficient autonomous agents.

1.1 Function

Biologists have thoroughly investigated functional questions using optimality approaches, introduced in Stephens and Krebs (1986). The attractiveness of such an approach comes from the tractable mathematics and a plausible natural connection between the currency that the biologist supposes the animal is optimising (eg. energy, offspring) and the ultimate biological goal function; the relative success of the animal and its offspring surviving in competition with all the other organisms. If we

could design agents that behaved optimally with respect to the constraints we place upon them, such as achieving tasks we wish to see completed, then we cannot expect to obtain better performance from any physical agent. There are, however, two problems with the optimality approach.

The first is the requirement of finding an appropriate utility function. For biologists, this is not such a serious problem; they can closely observe an animal's behaviour and determine a plausible utility dependent constraint, for instance the quantity of plankton caught in a caddisfly net (Georgian and Wallace 1981). McFarland and Bosser (1993) suggest possible utility functions for autonomous agents based upon satisfying the owner as opposed to just simply surviving. However, any such utility function must, at best, be only an approximation of what is actually 'good' for the agent and, at worst, can fail to capture the key features necessary to reward appropriate behaviour[1].

The second, and perhaps more serious, problem when considering applying optimality to agent design is that there is not any rote method of obtaining a mechanism for generating behaviour from a utility function. Methods available for calculating behaviour from a given utility function often require a perfect model of the environment and a goal state to work towards (Bellman, 1957) both of which are unavailable to an autonomous agent in an unpredictable world[2].

1.2 Mechanism

There have been a great many models developed by ethologists that have a mechanistic bent. The models of behaviour fall into two main camps either: sweepingly general, proposing only the highest level of behaviour selection (eg. Baerends, 1976; Dawkins, 1976; McFarland and Houston, 1981); or highly specific, considering only a small but pertinent feature of behaviour selection observed in animals (eg. Lorenz, 1950; Ludlow, 1976). Although the theoretical design of a mechanism for an animal's behaviour selection is a comparable problem to that of AI's action selection problem, biologists consider it to be a hard problem because of the complexity of animal behaviour sequencing. General accounts are not precise enough to implement (Tyrrell, 1993a) and specific accounts deal with problems that are not of any general use for the design of an animat's brain (Webb, 1994).

[1]The implication of contending that a utility function entirely captures an agent's behaviour is either that the utility function has encoded within it an entire world model; or that the world is conducive to being approximated by simplification methods such as symmetry.

[2]Such an approach to modelling is adopted by economists and most other rationalist methods of behaviour selection; for instance, planning in classical AI.

1.3 Approach

Notwithstanding these precautionary words, we would also contend that there is much that can be drawn from taking the same perspective as an ethologist in approaching the design of animats. This paper attempts to demonstrate a design approach that satisfies both the functional and mechanistic accounts of behaviour. In addition, the model presented below is based on a motivation-based control system that takes behaviour selection one step beyond that of general system selection method (Spier and McFarland, date), to incorporate a further level in what might be the hiearchical decomposition of behaviour.

2 The two resource problem

The two resource problem provides the minimal scenario for a decision making algorithm (Spier and McFarland, date). In the case of animals, such resource pairs could be food and water (Toates, 1980) or money and play (Cabernac, 1992); in the case of animats the pair could be energy and work (McFarland and Spier, 1996). The task of the agent is to control its behaviour so that it never lets any (either) of its essential state variables reach a lethal limit.

2.1 The agent as a state space

The state space model for agents was proposed by Sibly and McFarland (1974) and further developed in McFarland and Sibly (1975) and McFarland and Houston (1981). Within this framework an agent is characterised as a minimal set of internal state variables that can completely describe its physiological state. These variables sit within a Euclidean vector space as its orthogonal axes. In this space there will be regions that the agent can never encounter, for instance negative hormonal levels, and regions that should the agent cross into then it would die. The boundary of regions that are fatal to the agent are called lethal limits. Figure 1 illustrates such a state space. The task of the agent within such a model is to maintain 'homeostasis' of its state variables under perturbations from its own behaviour (eating makes you thirsty) and the environment (at night it gets colder).

2.2 The quadratic cost function model of the two resource problem

Sibly and McFarland (1976) considered the decision making process in the two resource problem from a functional perspective by postulating a quadratic cost function associated with a state space model of an animal possessing two essential state variables, hunger and thirst. They formulated the model in control theory and used Pontygarin's maximisation principle (Dixit, 1976 presents a clear development) to obtain the optimal behaviour. The

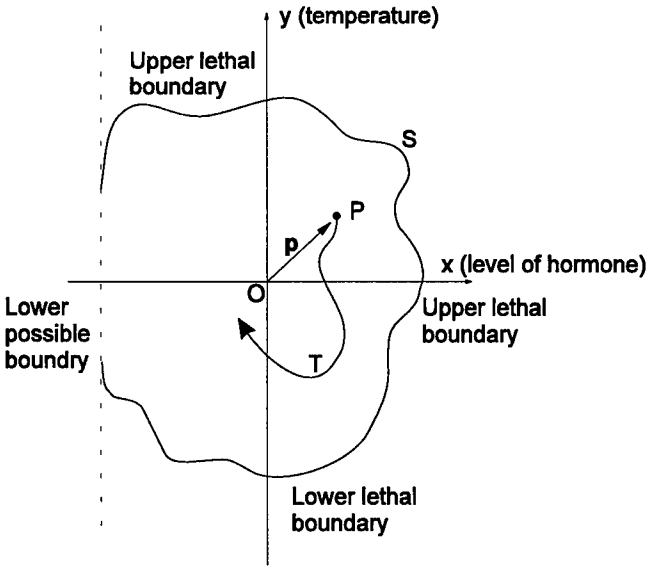


Figure 1: A state space. An example of a hypothetical two-dimensional physiological space with origin O corresponding to optimal body temperature and hormone level. The current physiological state is indicated by the point P or vector **p**. The boundary surface S separates the possible and lethal limits to values in the state space. T shows a possible trajectory the agent could take. After Sibly and McFarland (1974).

formulation was

$$\dot{h} = -r_h u_h,\ \dot{t} = -r_t u_t;\ 0 \leq \frac{u_h}{k_h} + \frac{u_t}{k_t} \leq 1 \qquad (1)$$

where h represents food deficit or hunger, t represents water deficit or thirst, the r_h and r_t are rate parameters, the u_h and u_t are the control variables that the agent has to set to perform the behaviours appropriate to reducing their respective deficits and the k_h and k_t are constraints upon u_h and u_t respectively which represent the maximum rate at which the behaviour can be performed. The r_h, r_t, k_h and k_t provide the model of the environment where r_h and r_t, called the availability, can be considered to be the density of the resource in the environment and k_h and k_t, called the accessibility, represent the ease with which the agent can obtain the resource though its behaviour (McFarland and Houston, 1981, Tovish, 1982). With such a model the cost function was hypothesised to be [3]

$$C = h^2 + t^2. \qquad (2)$$

[3]The actual cost function used in Sibly and McFarland (1976) was $C = h^2 + t^2 + u_h^2 + u_t^2$, which nicely predicts the negative exponential satiation observed in operant experiments (McCleery, 1977), however, as noted in that paper, when h and t are much larger than k_h and k_t then the approximation $C = h^2 + t^2$ holds except for when the agent is near satiation.

This can be justified using geometric common sense by noting that as a deficit gets larger the cost of possessing that deficit increases at greater than a linear rate (however Houston and McFarland (1980) provide a more comprehensive justification), although the choice of a quadratic function over any other convex function was made purely for mathematical simplicity (McCleery, 1977). This cost function has the appropriate property that the cost increases the further away from the homeostatic equilibrium point the animal's essential state variables lie. Sibly and McFarland (1976) calculated the solution to this problem using Pontryagin's maximisation method and found it could be summarised as

$$\begin{aligned} &\text{if } h.r_h.k_h > t.r_t.k_t \text{ then eat,} \\ &\text{if } h.r_h.k_h < t.r_t.k_t \text{ then drink;} \end{aligned} \qquad (3)$$

this solution combines the agent's state with the parameters that describe the environment.

2.3 *A system-level motivation model*

If we consider that the cues an agent receives from the environment about various resources allow it to assess the potential rates of gain or loss to its state variables then we can, say, associate the agent's perceived food cue with the $r_h.k_h$ in the quadratic cost function model. This is very attractive because under the assumption that the cue predicts a rate of gain of resource, we then can hypothesise that multiplying together the deficit and cue of any system provides a motivational 'strength' for that system to act as a common currency (McFarland and Sibly, 1975) in a behaviour selection system. Sibly (1975) showed that Barbary doves satisfied such a rule when placed in operant conditions under food and water deprivation; McFarland and Spier (1996) showed that such a rule generated plausible and appropriate behaviour in both a simulated agent and a real robot.

2.4 *Availability and accessibility*

In **2.2** the resources in the environment are modelled by two parameters, the availability r and the accessibility k. At first consideration this dichotomy may seem arbitrary, and r and k should be conflated into one variable. However, these parameters provide a powerful way with which to consider the environment. The availability is associated with the density of the resource in the environment. Such a density can be manifested in many ways. For instance, water could be in small units but uniformly distributed over the world, much like dew, or food could be in large units but at a low density, much like insect prey. Both alternatives could yield the same global density of resource in the environment.

Accessibility is associated with the ease with which an agent can obtain the resource through its behaviour.

Consider the scenario of an agent trying to find nuts underneath a carpet of leaves in a garden. A bird may peck at individual leaves, lifting them up, to inspect for nuts; such a behaviour involves great effort but does not yield many nuts. Consequently this behaviour would have a low k. An animat may have a vacuum attachment which sucks up the leaves and nuts, spitting out the leaves and caching the nuts, such a behaviour would have a high k (although the energy consumption of such an agent would be much higher than that of the bird with its low k).

2.5 Optimal tool selection

Any tool that the agent uses can be considered to possess an accessibility value. For instance, continuing with the nut example, the agent could crack the nut with its own hands or use a nut cracker. Both operations, crack nut with hands and crack nut with nut cracker have k values and both are tools. However it would be expected that the k for cracking a nut by hand is lower than the k obtained by using a nut cracker. Any group of task-achieving behaviours can be considered to be a tool, any task may be achieved with many alternative tools (McFarland and Bosser, 1993). Examples of possible tools from this perspective are: eating, sweeping with a brush, opening a door, using a map (internal or external) to get to the restaurant, or, even, the use of a plan to obtain a future consummatory behaviour.

By attributing to each tool its own accessibility value with respect to each resource we can quantify the relative advantage of using one tool as opposed to another. Here the real attractiveness of the r, k dichotomy reveals itself. The task of the behaviour selection system would be to select which tool to use as opposed to which system to satisfy (cf. **2.3**).

Referring back to the model (1) in **2.2** and using the same cost function, the optimal solution under these new definitions remains at

$$\begin{aligned} \text{if } h.r_h.k_h &> t.r_t.k_t \text{ then eat,} \\ \text{if } h.r_h.k_h &< t.r_t.k_t \text{ then drink.} \end{aligned}$$

However the constants k_h and k_t are not fixed values describing the environment but variable depending upon which tool the agent is considering using. The optimal behaviour at any time would then be to use the tool with the associated k that has the highest $d.r.k$ product, where d is the deficit of a state variable, r is the global density of resource relevant to the state variable in the environment and k is the accessibility value of the tool under consideration with respect to the resource[4].

[4]Although in this formulation of the $d.r.k$ model scalar values have been used, it does not take much extra work to show that vector r and k complexes can also be considered where, as with d, each component corresponds to an essential state variable.

3 A finer-grained motivation based model

Having carefully defined accessibility and availability it is an easy step to move from the functional behaviour predicting model to a mechanistic version. Following the methodology used to find the system-level behaviour sequencing algorithm in **2.3** we can associate the d, r and k of the optimal model to entities in a mechanistic model. As before, we associate a d with the deficit of each essential state variable and a r_d with the cue from the environment about each resource (where the d ranges over all the state variables for each resource). The new feature of this model is to associate a k_r to the cue from the environment regarding each tool (where the r indexes the performance of the tool with respect to the particular resource).

The behaviour selection algorithm would then simply be to evaluate every $d.r_d.k_r$ possibility and, as such, assign a motivational strength to each possible option of using a tool to acquire a resource. At any time the complex with the highest value is selected, the resource and tool used to form it specify how the agent should then behave.

To provide a more concrete example consider the scenario of when an agent needs to maintain its food and water levels. Within the environment there are local distributions of nuts and shallow trays of water. Also within the environment are two physical tools in the form of rocks and cups. Now, there are two deficits with which to associate a d, hunger (h) and thirst (t) and two resources, nuts (r_h) and water (r_t). Less intuitively, there are six tools: cracking and eating nuts with the agent's forelimbs ($k_{h,agent}$), cracking and eating nuts with the rock ($k_{h,rock}$), drinking water from the shallow tray ($k_{t,agent}$), pouring the water into the cup and drinking from it ($k_{t,cup}$), drinking water with a rock ($k_{t,rock}$) and cracking nuts with a cup ($k_{h,cup}$). Of course, the k value for the last two described tools will be rather low compared to the initial four and it would be expected that the k value for the appropriate tool use would be higher than the k value of consuming a resource without any tool at all.

4 Simulation

Sections **2** and **3** outlined the theoretical functional and mechanistic arguments for a tool level motivation based model. The simulation experiment described below compares the performance of an agent with an implementation of the $d.r.k$ motivation model against agents whose control systems are not so developed. The simulated world was closely based upon the rock and cup example discussed in section **3.**

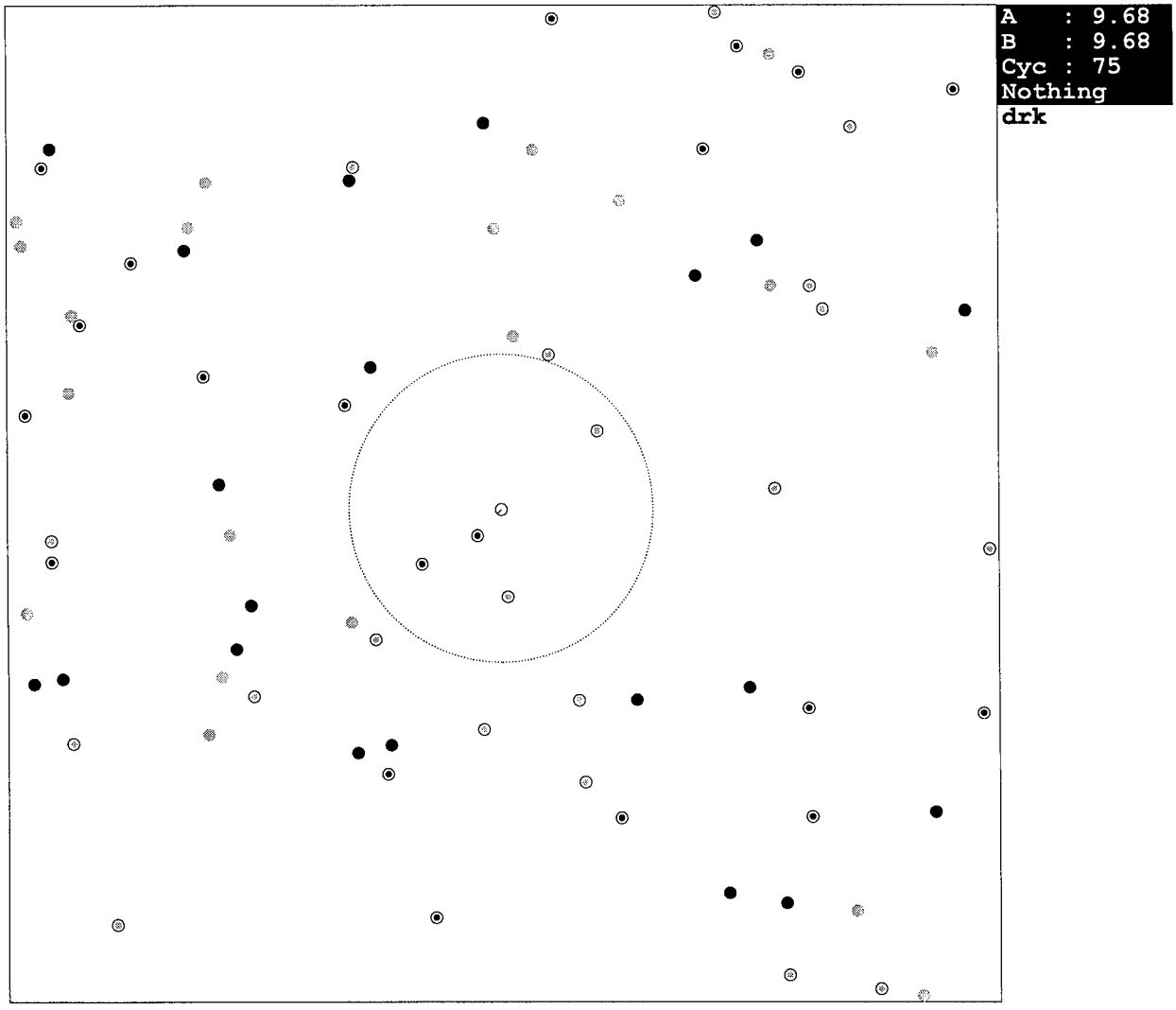


Figure 2: A screen dump captured during an agent trial within the 10000×10000 unit simulation environment. The agent can be found in the centre encircled by a typical extent of its range sensor, the two types of tools and resources are uniformly distributed in the environment. Tools are denoted by a half filled circle in the shade of their respective resource. The right-hand side window shows the agent's current state, lifetime, held tool and control strategy employed. The agent can detect the nearest objeect of each type within its sensory range.

4.1 Environment

The simulation environment consisted of a continuous torrodial surface of 10000×10000 units within which resided the agent, two types of resource and two types of tool. The resources and tools were of 60 units radius and randomly distributed on the surface. The agent was manifested as a circle of radius 60 units and possessed sensors that could identify the bearing and a normalised distance to the nearest resource and tool of each type within the maximum range of the sensors[5] (figure 2). The agent contained two internal state variables (A and B). If through its actions it permitted either to fall to zero then it would die. Following the example in section **3**, these variables could be considered to be food and water and the tools considered to be a rock and a cup. While the agent was active it expended both of its resource state variables at a fixed ratio. When the agent was adjacent to either of the two types of resource then it could consume them to increase their allied state variable at the cost of a smaller loss to the other internal state variable, this representing the time and/or effort used in the handling the resource. Should the agent be holding a tool, then it could opt to use the tool to con-

[5] Although the sensors returned perfect information within their sensitivity range, it should be noted that these models are not susceptible to noise in the same way as possibly more brittle ones. Since the value of each motivation system is recalculated each cycle, any noise would be ignored in the long term for the same reasons that these models can exhibit opportunism (Spier and McFarland, date).

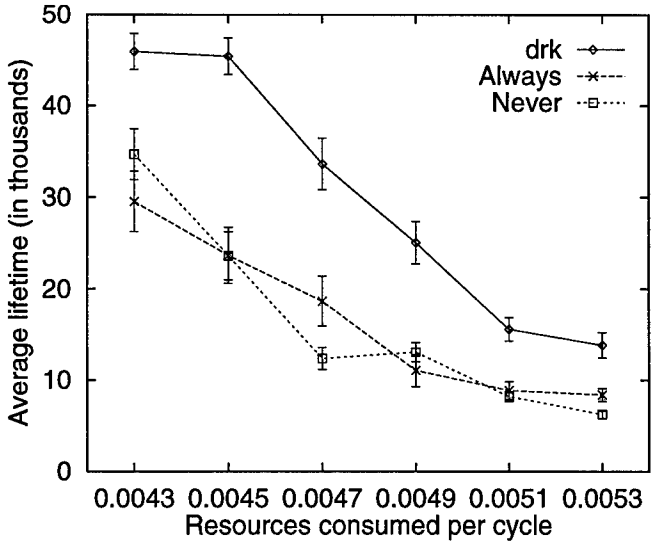


Figure 3: Results comparing the three behaviour strategies in an environment with 20 of each type of resource or tool and where the accessibility of the tools was twice the nominal accessibility. The maximum sensory range was a radius of 1500 units around the agent. Each point represents the mean of 40 trials bracketed by its standard error. ($r_A = 1, r_B = 1, k_{A,rock} = 2, k_{A,cup} = 0.5, k_{B,rock} = 0.5, k_{B,cup} = 2, k_{A,agent} = 1, k_{B,agent} = 1$)

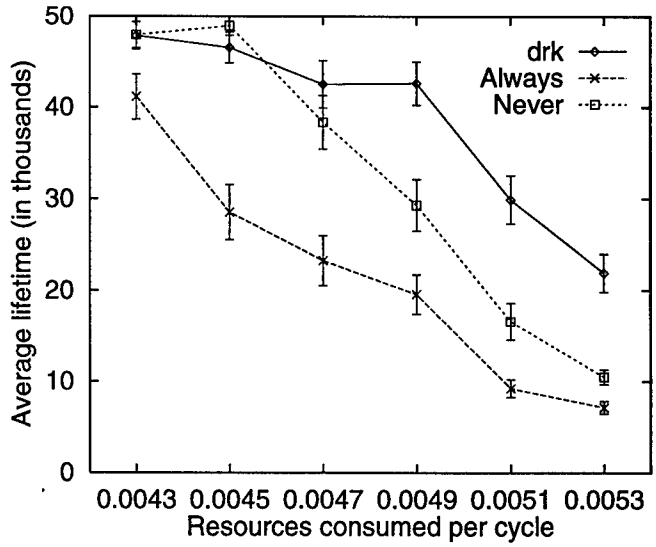


Figure 4: Results comparing the three behaviour strategies in an environment with 20 of each type of resource, 40 of each type of tool and where the accessibility of the tools was 1.5 of the nominal accessibility. The maximum sensory range was a radius of 2000 units around the agent. Each point represents the mean of 40 trials bracketed by its standard error. ($r_A = 1, r_B = 1, k_{A,rock} = 1.5, k_{A,cup} = 0.5, k_{B,rock} = 0.5, k_{B,cup} = 1.5, k_{A,agent} = 1, k_{B,agent} = 1$)

sume the resource, though it need not. The payoffs for each resource type were symmetric and when a resource was consumed it was randomly replaced in the environment using a uniform distribution. Likewise, should the agent be adjacent to a tool then it could opt to pick it up. The agent was only allowed to hold one tool at any time but could swap its currently held tool for an adjacent one. After the tool was used to consume a resource it was then randomly replaced in the environment.

The task of the agent was to survive. For measurement purposes, the agent's opportunity to exist in the environment was curtailed at 50,000 cycles. A cycle consisted of: moving a standard distance in any chosen direction; should the agent be adjacent to a resource, consuming the adjacent resource of its choice; or, picking up a tool, if the agent was adjacent to it.

Strategies tested within this environment were compared by slowly making the environment harder in which to exist. This was performed in two ways, the strategies were titrated either by varying the quantity of each state variable expended per cycle or changing the maximum range that the agent's sensors could resolve. Appendix 1 contains a list of all the essential agent and environment determining constants.

4.2 Decision Strategies

The tool-level motivation-based model (fully described in Appendix 2) was compared against two other possible models. The first was a system-level motivation-based model that ignored the tools and simply consumed the resources following an algorithm based on the model described in **2.3**. The second, while based upon a system-level motivational-model, tried to always use tools by following a simple although carefully considered set of rules (See Appendix 3 for a complete description).

4.3 Results

Figures 3 and 4 contain illustrative examples of the results obtained from the simulation, each point represents the mean of 40 trials. Figure 3 shows the performance of the three strategies in a simulated environment in which there were 20 tools of each type and each tool, upon appropriate use, doubled the resource gain. The *d.r.k* strategy clearly performed the best over the whole range of agent's resource expenditure per cycles tested. In addition, the system-level strategy performed comparably to the strategy that always tried to use tools. Figure 4 shows the performance of the strategies in a slightly different environment where the range sensors' radius was increased by a third (hence increasing the amount of information about the world that the agent posesses), the

quantity of tools available for each resource was doubled and the accessibility of the tools halved (resulting in the same $r.k$ value, as before, for the global density). Here, at low resource expenditures per cycle, the system level strategy and the $d.r.k$ strategy performed comparably whereas the 'always' strategy performed uniformly worse. At higher values of the agent's resource expenditure per cycle, when surviving in the environment was harder, the $d.r.k$ strategy and the system level strategy separate out with the $d.r.k$ strategy outperforming the system level strategy.

4.4 Analysis

Clearly the $d.r.k$ model has performed much better over a wider range of environments than the two chosen controls. It is worth considering why this is the case. All models exhibit opportunism and motivational system trading-off at least at the system-level (Spier and McFarland, date), so for the $d.r.k$ model to have outperformed the other two strategies it must be exhibiting opportunism in regard to whether it uses the tools or not. This opportunism can be further broken down into two components. The first is simply ignoring far away but appropriate tools if the agent is close to a resource; this effect can be seen in the underperformance of the 'always' strategy in an environment in which there were a relatively large number of low accessibility tools. The second is that the agent will preferentially hold the tool, even by a swap, associated with the system of the current highest motivational value.

The significance of this can be illuminated by an example. Consider the scenario of an agent who is more thirsty than hungry. The agent is carrying a cup but no resources are in sight. The agent encounters a nut and a rock in close proximity. The $d.r.k$ model will head for the rock (since the, as yet unobtained but high rock cue is larger than the agent's nominal accessibility value for the nut, and the hunger motivational system, responding to the food cues, will now be higher than that of the thirst system). Dropping the cup the agent will then pick up the rock, consume the nut and (since there are no objects save the cup in the sensor range so the thirst motivational system will have returned to its formerly superlative state) pick up the cup again. From the observers point of view, this behaviour sequence is tantamount to appetitive planning behaviour in which the agent notices an opportunity, so, rebuilds the plan incorporating the new information obtained from the world. Such behaviour sequences were not uncommonly exhibited in the simulation by the $d.r.k$ strategy, and, of course, no such plan building occurred, the agent needed only to respond to stimuli in the environment.

5 Conclusion

Though it is noted that ideas in ethology have not been developed enough to provide realisable solutions to the problems in behaviour that need to be resolved for successful construction of animats, they still can supply powerful intuitions. A carefully considered approach, adopting from ethology, can provide insights into possible animat mechanisms, while additionally furthering a functional justification for the mechanisms as opposed to the more arbitrary, engineering type, justification by way of performance satisfaction. For ethology, it should be noted that animat research forces one to consider how behaviour is to be implemented; a question involving many lessons worth learning. The model developed and extended in this paper demonstrates that when designing animat 'brains' it may not be necessary, and certainly not obligatory, to use a form of rationality that admits reasoning to obtain appropriate behaviour sequences. Rather, at least some behaviour for which planning could be considered necessary, may simply be based upon a sophisticated motivational system, trading off between opportunities in response to the environment. Of course, the question remains, how far may such approaches, like the model discussed, extend into the realms for which some think rational planning to be necessary?

Appendices

Appendix 1

The parameters used to describe the environment (with the values used in brackets) were : world size (10000×10000), r_A (1), r_B (1), $k_{A,rock}$ (2), $k_{A,cup}$ (0.5), $k_{B,rock}$ (0.5), $k_{B,cup}$ (2), $k_{A,agent}$ (1), $k_{B,agent}$ (1), maximum distance the sensors can resolve (1500), state A loss per cycle (0.0045), state B loss per cycle (0.0045), decrease in internal state B after consumption of A (0.2), decrease in internal state B after consumption of A (0.2), A items in the environment (20), B items in the environment (20), Rocks in the environment (20), Cups in the environment (20), Ambient cue value (0.2), distance the agent moves in one cycle (10), maximum value of internal state A (40), maximum value of internal state B (40), initial internal state A (10), initial internal state B (10), A item radius (60), B item radius (60), agent radius (60), tool radius (60).

Appendix 2

Decision rules for the $d.r.k$ strategy. Every cycle the cue values of resources A (r_h), B (r_t) and tools Rock (q_{rock}), Cup (q_{cup}) were evaluated as the maximum of either a normalised [0,1] cue value within the maximum sensor range or an ambient cue value (McFarland and Spier, 1996). Six motivation complexes were formed : $h.r_h.k_{h,agent}$, $h.r_h.(q_{rock}.k_{h,rock})$, $h.r_h.(q_{cup}.k_{h,cup})$,

$t.r_t.k_{t,agent}$, $t.r_t.(q_{rock}.k_{t,rock})$ and $t.r_t.(q_{cup}.k_{t,cup})$. (In this simulation, r_h and r_t were considered orthogonal.) The behaviour appropriate to the tool specified in the highest motivation complex was performed for that cycle; if the agent was not in possession of the tool, it would approach it, otherwise it would approach the resource specified in the complex.

Appendix 3

Decision rules for the 'always' strategy. In order of precedence :

1) If the agent can see no resources but can see a tool, then approach or obtain the tool, with preference for a tool associated with the most significant motivational system.
2) If the agent is near a motvationally significant resource, not holding a tool and also near the appropriate tool, head for or obtain the tool.
3) If the agent is holding the appropriate tool for the selected system, then approach and consume.
4) If the agent is holding nothing or an inappropriate tool for the selected system, then consume the resource anyway but do not use a tool.
5) Move forward with a small probability of making a deviation.

As with all the strategies, these rules were evaluated afresh every cycle.

References

Agre, P. and Chapman, D. (1987). Pengi: An implementation of a theory of activity. In (Eds.), *Proceedings of the Sixth National Conference on Artificial Intelligence, AAAI-87.* Morgan Kaufmann.

Baerends, G.P. (1976). The Functional Organization of Behaviour. *Animal Behaviour*, **24**, 726-738.

Bellman, R.E. (1957). *Dynamic Programming.* Princeton,NJ, Princeton University Press.

Blumberg, B. (1994). Action Selection in Hamsterdam: Lessons from Ethology. In Cliff, D., Husbands, P., Meyer, J.-A. and Wilson, S. (Eds.), *From Animals to Animats 3: Proceedings of the third International Conference on Simulation of Adaptive Behavior*, pp. 108-117. MIT Press, Cambridge, MA.

Brooks, R.A. (1990). Elephants Don't Play Chess. *Robotics and Autonomous Systems*, **6**, 3-15.

Cabanac, M. (1992). Pleasure: the Common Currency. *Journal of Theoretical Biology*, **155**, 173–200.

Dixit, A.K. (1976). *Optimization in Economics.* Oxford University Press, Oxford.

Dawkins, R. (1976). Hierarchical organisation: A candidate principle for ethology. In *Growing Points in Ethology* (ed. Bateson, P. and Hinde, R.). Cambridge: CUP.

Georgian, T. and Wallace, J.B. (1981). A model of seston capture by net-spinning caddisflies. *Oikos*, **36**, 147-157.

Halperin, J.R.P. (1991). Machine motivation. In Meyer, J.-A. and Wilson, S. (Eds.), *From Animals to Animats: Proceedings of the First International Conference on Simulation of Adaptive Behavior.* MIT Press.

Houston, A.I. and McFarland, D.J. (1980). Behavioural Resilience and its Relation to Demand Functions. In *Limits to Action: The Allocation of Individual Behaviour* (ed. Staddon, J. E. R.), pp. 177-203. New York: Academic Press.

Lorenz, K. (1950). The comparative method in studying innate behaviour patterns. *Symposia of the Society of Experimental Biology*, **4**, 221-268.

Ludlow, A. (1976). The behaviour of a model animal. *Behaviour*, **58** , 131-172.

Maes, P. (1990). Situated Agents Can Have Goals. *Robotics and Autonomous Systems*, **6**, 49-70.

McCleery, R.H. (1977). On Satiation Curves. *Animal Behaviour*, **25** , 1005-1015.

McFarland, D.J. and Bosser, T. (1993). *Intelligent Behaviour in Animals and Robots.* Cambridge, MA., MIT Press.

McFarland, D.J. and Houston, A.I. (1981). *Quantitative Ethology : The State Space Approach.* London, Pitman.

McFarland, D.J. and Spier, E. (1996). Basic cycles, utility and opportunism in self-sufficient mobile robots. *Robotics and Autonomous systems*: Forthcoming special issue from the Monta Verita conference on Autonomous Agents.

McFarland, D.J. and Sibly, R.M. (1975). The Behavioural Final Common Path. *Philosophical Transactions of the Royal Society (Series B)*, **270**, 265–293.

Rosenblatt, K. and Payton, D. (1989). A Fine-Grained Alternative to the Subsumption Architecture for Mobile Robot Control. In (Eds.), *Proceedings of the IEEE/INNS International Joint Conference on Neural Networks.* IEEE.

Sibly, R. (1975). How incentive and deficit determine feeding tendency. *Animal Behaviour*, **23**, 437-446.

Sibly, R.M. and McFarland, D.J. (1974). A State Space Approach to Motivation. In *Motivational Control Systems Analysis* (ed. McFarland, D. J.), pp. 213–250. London and New York: Academic Press.

Sibly, R.M. and McFarland, D.J. (1976). On the Fitness of Behaviour Sequences. *American Naturalist*, **110**, 601–617.

Spier, E. and McFarland, D. (date). Possibly Optimal Decision Making under Self-sufficiency and Autonomy. Submitted to *Adaptive Behavior.*

Stephens, D.W. and Krebs, J.R. (1986). *Foraging Theory.* Princeton, NJ, Princeton University Press.

Tinbergen, N. (1951). *The Study of Instinct.* Oxford, Clarendon.

Toates, F. (1980). *Animal Behaviour - A Systems Approach.* New York, John Wiley

Tovish, A. (1982). Learning to improve the availability and accessibility of resources. In *Functional Ontogeny* (ed. McFarland, D.). London: Pitman books.

Tyrrell, T. (1993a). *Computational Mechanisms for Action Selection.* PhD thesis, Cognitive Science Department, University of Edinburgh.

Tyrrell, T. (1993b). The Use of Hierarchies for Action Selection. In Meyer, J.-A., Roitblat, H. L. and Wilson, S. (Eds.), *From Animals to Animats 2: Proceedings of the Second International Conference on the Simulation of Adaptive Behavior.* MIT Press.

Webb, B. (1994). Robotic Experiments in Cricket Phonotaxis. In Cliff, D., Husbands, P., Meyer, J.-A. and Wilson, S. (Eds.), *From Animals to Animats 3: Proceedings of the third International Conference on Simulation of Adaptive Behavior.* MIT Press/Bradford Books, Cambridge, MA.

Wilson, S.W. (1991). The Animat Path to AI. In Meyer, J.-A. and Wilson, S. (Eds.), *From Animals to Animats: Proceedings of the First International Conference on Simulation of Adaptive Behavior.* MIT Press.

What Are Emotions For?
Commitments Management and Regulation Within Animals/Animats Encounters

Michel Aubé
Université de Sherbrooke,
2500 Blvd Université,
Sherbrooke, Qc, CANADA, J1K 2R1
Phone: 1 (819) 821-7426
Fax: 1 (819) 821-7950
maube@courrier.usherb.ca

Alain Senteni
Université de Montréal,
C.P. 6128, succursale A,
Montréal, Qc, CANADA, H3C 3J7
Phone: 1 (514) 735-9826
Fax: 1 (514) 735-9826
senteni@iro.umontreal.ca

Abstract

The hypothesis that emotional mechanisms are wired in the repertoire of higher species is not as new as is its impact on the nature of autonomous agents underlying animats' design. In an evolutionary perspective, it looks as if, the more rational a species gets to be, the more emotional it happens to be as well. Thus, the study of emotional behavior raises the question of adaptiveness: what are emotions for, in terms of benefits for the individual? This paper attempts to answer such questions by proposing a model that links autonomy to motivation, to resource management and to emotions. The model distinguishes two kinds of resources: first-order ones, directly consumable, and second-order ones whose access is only guaranteed by a mediating agent, committed to give them. The model defines needs (such as those associated with states of hunger or thirst) as motivational processes insuring management and regulation of directly consumable first-order resources, while emotions (such as anger or fear) appear as motivational processes insuring management and regulation of second-order resources - namely commitments - within complex species for which they become the most important of all resources, and the key for achieving multiple goals. The paper concludes that emotions, like needs, are powerful control structures that have been designed in to protect agents and societies of agents from running out of resources. Such mechanisms meet some essential requirements that autonomous agents are faced with.

1. Introduction: a short story by K. Lorenz

> "I have talked before about Bully, my Bulldog. He was already old, yet still brisk when I got hold of Hirschmann, the Hanover Bloodhound - or should I say when this one took possession of me. His arrival was a rude blow for poor Bully, and if I could have foreseen how much he would suffer from jealousy, I probably would have resisted Hirschmann's seduction. The atmosphere was tense for days until exploded one of the most violent dogs' fights I have ever witnessed [...] As I was trying to separate the fighters, Bully bit my finger accidentally. It was the end of the fight, but poor Bully had been struck with the greatest shock a dog's nervous system could ever register. He fell into depression, and although I did not punish him, but to the contrary caressed and fondled him, he remained paralysed down on the carpet, a round ball of despair unable even to get up [...] It was days before he started eating again, and even then we had to feed him from hand. For weeks, he would come to me only with an attitude of humble supplication, in total breach with his usual temper of a fierce dog that had never been servile. Now the interesting thing [is that this dog] had never bitten anyone before, and hence had never been punished for such a fault." [Lorenz 70, p. 232-235 - the authors' translation]

Although filled with much anthropomorphism, this brief account of Bully's fate certainly does not sound too unrealistic to anyone who has shared a pet's life for some significant amount of time. The interactions thus experienced, *within and across species,* seem loaded with the emotional flavours of joy, attachment, anger, sadness, jealousy, and even, as suggested here... guilt! And as Lorenz suggests in the punch line, there is good reason to believe that Bully's complex reaction has not been conditionned, but is a natural one to the situation, already built into the animal's repertoire.

To the behavioral scientist, one obvious question is why should it be so, what usefulness lies behind those reactions? In other words, emotional behavior raises the question of *adaptiveness:* what profit has been gained through evolution for members of a species to feel joyful or angry or fearful or guilty, what is the added survival value over individuals belonging to species - typically simpler and evolutionary older ones - that do not ever get emotional? Are there any systematic reasons for the fact that the more rational a species gets to be, the more emotional it seems to get as well?

The main purpose of this paper is to propose a coherent line of explanation of emotional behavior in order to answer the questions stated above. This attempt relies upon concepts borrowed from Artificial Life (ALife) and

Distributed Artificial Intelligence (DAI) and is part of an ongoing project of using emotional structures as a metaphor from which to specify powerful control structures for Multi-Agents Systems (MAS) [Aubé & Senteni 95] [Aubé & Senteni 96]. An additional aim is to stress the theoretical importance of the motivational aspect of behavior and to show how this dimension could profitably be integrated with the current theorizing about rationality and goal-oriented behaviors within cognitive science.

The second section of the paper enumerates some of the requirements autonomous agents are to be faced with. The third one looks at motivation as a prerequesite for autonomy, links autonomy to resource management, and sketches two layers of control that developed on top of one another, in order to insure such management. In the following section, emotions as theoretical constructs are then deliberately restricted to the social domain, as some kind of interactional devices, that act more precisely as *commitments operators* between individuals to regulate the additional resources (termed second-order here) that other agents may provide to one another. It also considers two special kinds of long term commitments, affiliation and status, that bear consequences upon the postulated design. Future works are finally forecast as a conclusion.

2. Requirements for Autonomous Agents

If one is to understand the architecture of living organisms so as to design artificial agents that are to prove adaptive in complex environments, what requirements are these beings likely to be faced with for them to be robust, and what kind of ontology would be apt to meet such requirements?

One inescapable requirement is certainly that these agents will have to operate within *open systems* [Hewitt 85] [Hewitt 91] wherein continuous change is the rule, and where the predictability of important events could sometimes be rather low. Hence there is the need for a sufficient amount of reactivity to those external events. The behavior of such agents is also to be restricted to "arms' length relationships", with typically no possibility for a global view over all the relevant aspects of a problem or situation. This raises the need for negotiation, so as to obtain complementary pieces of information from other agents.

On the other hand, natural agents are to be "evolvable" creatures, submitted to the *subsumption constraint* [Brooks 91]. This means that they should not have been faced with too abrupt changes or global reorganization in their design. Hence their real-time reactive capabilities have to result from the operation of successive layers of control. So there is an argument in favor of the functional additivity of those interconnected layers, in such a way that, at each level, the design proves adaptive in itself, but that every successive layer adds new functionalities without disrupting the preceding ones.

Being immersed in open systems, these agents also reach for goals through *augmented planning* [Oatley 92]. They are indeed complex enough to have multiple goals of different kinds, of comparable priority, to be handled mostly in parallel. Yet they are faced with limited (at arms' length) knowledge and have to compete for scarce resources. And they have multiple agents to deal with, which raises issues of communication, cooperation, conflicts, negotiation, organization...

Being decentralized with encapsulated knowledge bases and embedded reactive behaviors, those agents have to display a fair amount of *autonomy* [Maes 91] [Castelfranchi 94]. But mere "executive autonomy" whereby one is only free to execute whatever command comes from outside, is clearly not sufficient here. Real autonomous agents should have the capability for self determination of goals. This means that, more often than not, they will set and pursue their own interests, with the consequence that *they could not always be assumed to be sincere or benevolent.* In pursuit of their goals, they will also have to influence other agents of their kind. This implies that, by virtue of sharing a common ontology, such agents should themselves be liable to dependence and influencing.

So the stakes are fairly high, and the challenge of conceptualizing agents that are to meet such requirements, even in the simplest possible case, is no easy task. One contention of this paper is that the underpinning of motivation provides one essential line of attack for this design problem.

3. Motivation is about Resource Management

In specifying "a principled theory of agency and autonomy", [d'Inverno & Luck 96] defined autonomous agents as progressive refinements within a hierarchy of computational entities: *Objects* are first defined as generic *Entities* with (declarative) attributes and (procedural) capabilities. From then on, *Agents* are seen as *Objects* with goals, and finally *AutonomousAgents* are conceptualized as *Agents* with motivations. These authors further defined motivation as "any desire or preference that can lead to the generation and adoption of goals and which affects the outcome of the reasoning or behavioral task intended to satisfy those goals" (p. 73).

Indeed, the concepts of autonomy and motivation are intimately related: if agents are ever to become autonomous, it will only be through getting control over their motivations, by having ways of setting their own goals. But what is motivation about in the first place or, as [Ferber 96] would put it, where do goals ultimately come from? Our claim is that motivation basically has to do with *managing resources,* a task which is most crucial and most vital for the agents, and which hence becomes their ultimate goal. Such thinking is indeed what made behavioural ecology [Krebs & Davies 87] so efficient in

explaining much of animal behavior, and it seems hard to imagine that artificial autonomous agents could profitably be designed otherwise!

The next step is hence to specify what is to be counted as a resource. In the agent context, we define resources more or less as energy is defined in physics: whatever enables - and is required for - the production of work. It could be physical energy like heat or electricity or food, but it could also be other things: time, money, affiliation, power, knowledge... For instance, if a robot has to get to some target point P, this should cost it some amount of physical energy, be it electricity or fuel. Now, if the animat learns about a shorter route than usual to P, this specific knowledge is clearly worth some economy in fuel: it could hence be counted as a resource, and even be quantitatively measured in terms of the energy saved. Likewise, power over other agents is a way of harnessing their resources to benefit one's own, and could as such be counted as a proper resource. Basically then, the concept of resource is tantamount to the *cost* of work or actions.

However it seems useful to refine the concept and to distinguish at least two different kinds of resources: first-order and second-order. First-order resources are directly consumable ones that an agent already has at his disposal (available food or water for animals, processing time or memory for computers). Second-order resources, on the other hand, are those that an agent gets only indirectly, through having other agents *committed* to give them to him: newborns can get access to food only through having their parents provide the resources for them. Hence, the basic distinction between first-order and second-order, is that the latter rests essentially upon a mediating agent. The apple one has at one's disposal for lunch is first-order, while a friend that said he would bring one is second-order. More specifically, it is not the agent *per se* that is the resource, but whatever makes it so that one can count on him: namely, *the commitment is the resource!*

It should also be clear that commitments are not mere "potential resources": some food across the river, that one could try to access, could be seen as such. But the definition of second-order resources proposed here relies upon some other agent, that one counts on. *It is intrinsically bounded with multi-agent societies.* This point is quite crucial: the commitment is a resource in mediating (insuring) access to some commodity, in a similar sense that a vehicle could become a resource by allowing access to some remote commodity. Now, this does not mean that commitments have to be formulated in a conscious or reflective way - attachment structures, for instance, do involve commitments between caretakers and infants that are presumably triggered through the activation of wired-in releasers and partially built-in structures [Reite & Field 85]. In niches of nurturing species, getting hold of the process of establishing and regulating commitments very likely became a requirement. Indeed, for newborns of those species, it is a life or death matter!

Now commitments have received various definitions in Distributed Artificial Intelligence (DAI). On the individualistic side, they are conceptualized as persistence in intention or goal pursuing [Cohen & Levesque, 90ab] [Dongha 94]. On the other extreme of social determinism, they are seen as an emergent property of agents being involved in many interlocking courses of action [Gerson 76] [Gasser 91]. And there is the middle-road interpretation of mutually agreed constraints (through promise or contract) on action, belief and world states [Winograd & Flores 86] [Bond 90]. These different definitions led in turn to various "translations" or implementations of the concept: as logical conditions for intentions to endure [Cohen & Levesque, 90ab] [Dongha 94]; as "agreed upon" actions, beliefs or goals [Winograd & Flores 86]; as production rules for the control of execution [Shoham 93]; as embodied within plans, conventions or social laws [Bond 90] [Jennings 93] [Shoham & Tennenholtz 95]; as quantified in terms of resources allocated to pursuing one's intentions [Gerson 76] [Bond 90] [Dongha 94].

The definition in terms of "persistence" seems too individualistic for planning agents embedded in "open" MAS. On the other hand, the more radical "emerging view" coming from social interactionists fails to offer concrete ways of representing how such interlocking gets bootstrapped in the first place within the agents' mental world. Our stance is that the middle-road view of mutually agreed constraints through promise or contract, could better meet the requirements, and even be translated into precise quantifiable terms. Commitments could indeed be measured in terms of allocation of resources and constraints upon resources [Gerson, 76] [Bond 90] [Dongha 94]. Typically, an agent A *commits* certain of its resources to another agent B (commits stands here for *does not give right away, but promises to give eventually*). As a consequence, B is *allocated* new resources, while A is *constrained* as to the usage of its own resources. For instance, A might be constrained not to consumate all of them and to take into account some amount that has to remain available to fulfill the commitment. On the other hand, such commitment could also involve for A setting the goal of acquiring for B some resources that A does not already possess.

Being specifically defined in terms of resources, commitments thus appear as one critical reason for the setting up of MAS: by regulating exchange, they enable the agents to use one another as a way to access additional resources that would otherwise remain out of reach, or too costly to obtain. This refers to what [Castelfranchi 90] has called the *Sociality* problem: "why does an autonomous agent enter into social relations" (p. 50) and is congenial with this author's proposed answer: "Sociality was invented to multiply agents' powers of achieving their goals (using the powers of other agents)" (p. 56). *Within complex species that pursue multiple goals, and who absolutely require as such the cooperation of other agents, commitments might well become the most important of all resources! One important corollary is that, MAS thus*

require an effective control mechanism to regulate and control commitment fluctuations. Hence, it is very likely that, in social species like ours, evolutionary forces would have favored and selected for precisely those organisms that would have become equipped with powerful mechanisms to insure control over commitments!

Within animals, *needs* - such as those associated with states of hunger, thirst, fatigue... - are the motivational processes [Toates 86] that insure management and regulation of directly consumable first-order resources, be it available food or water, rest and sleep, security or reproduction. In higher (social, or at least nurturing) animals, *emotions* - such as anger, fear, joy, sadness, guilt... - are the motivational processes that insure management and regulation of second-order resources, namely commitments. Very much like needs, emotions are powerful control structures that act so as to protect agents - and societies of agents - from running out of resources. In terms of the *subsumption* principle, they could be conceived as an additional layer of design that developed on top of the underneath layer of needs, using much of its control power, but taking over whenever the access to resources requires the cooperation of other agents. Figure 1 illustrates the operations of those two layers of control and their presumed interconnections.

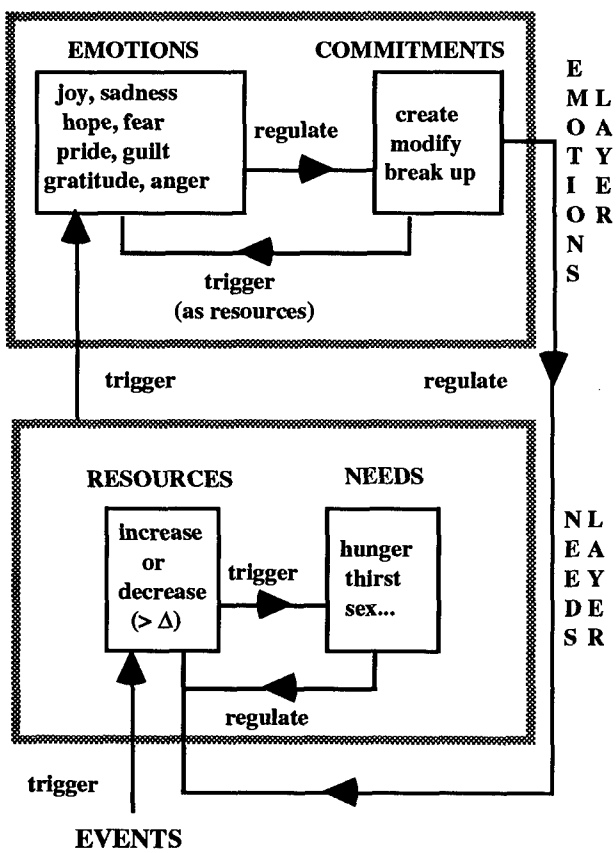


Fig. 1. Two layers of control for resource management.

As for needs, the emotional control processes are triggered by significant variations in the flow of resources, specifically second-order ones, for instance when there is a threat of breach in commitments or an opportunity for establishing new ones. Significant variations here means that the amount of increase or decrease of resources should exceed a certain predefined (although eventually ajustable) *threshold.* Certain events are likely to have an impact upon an agent's stock of resources. When those variations cross a critical threshold, needs are called in so as to regulate them. In more complex (social or nurturing) animals, other agents can supply for the missing resources, provided they are committed to behave so. Emotions are designed in to regulate the operation of this new layer.

Now, having restricted emotions to being commitments operators, leads us to distinguish between different motivational reactions, that we would not count as "emotions proper". For instance, the escaping reactions of lower organisms like fish, amphibians and most of the reptiles, that happen not to have evolved limbic structures [MacLean 93], are thought to rest upon a more primitive layer for handling needs (security being involved here). Our basic criterion for counting a motivational process as an emotion, is that it should provide for some communicative means in the service of interagent binding (such as alarm or distress calls, warning growls, tail waving...), even if the corresponding message is not actually sent in some particular situation.

In the case of a human reacting with a jump to a loud noise, or stepping aside in order to avoid a speeding car, we would say that his "fear reaction" is more complex (subsumptionwise), basically exploiting simpler old structures, but also profitting from the "additional higher" possibility of yelling as well, and thus expressing distress (i.e. calling for someone committed to help, such as a mother) or anger (i.e. warning that the transgressor should retract and amend). There is much experimental data from the psychophysiology of emotions that provide some suggestive evidence for different origins behind the repertoire of human fears [Öhman 86] and for the fact that some of these behaviors rest upon much simpler and evolutionary older structures [Ekman, Friesen & Simons 85]. *It is our contention that reserving "emotions" for the control structures that are responsible for commitments management will do much to help disambiguate the term itself and specify the possible design of these structures.*

4. The Emotion Engine

Now let us sketch how this emotion layer could operate. First we must specify the *SIGNALS* which are going to trigger the regulating processes. Just as needs are activated when variations in first-order resources pass beyond some threshold [Toates 86], emotions are triggered by significant gains or losses in commitments (second-order resources), that could be evaluated numerically in terms of the corresponding first-order resources they guarantee access to.

This would lead to positive and negative emotions respectively. Positive emotions act so as to reinvest the gain (that triggered them in the first place) in order to buy access to further resources (e.g. establish new commitments): they operate as accumulators or *amplifiers*. Negative emotions, on the other hand, are used so as to fill the breach, slow down the running waste, and eventually to call for help so as to replace the incurred loss: they operate as *regulators*.

As to formulate proposals for the operation of the emotional processes themselves, Herbert Simon was the first scientist within the AI community, to dare make an offer, back in the late sixties: "If real-time needs are to be met, then provision must be made for an interrupt system [...] all the evidence points to a close connection between the operation of the interrupt system and much of what is usually called emotional behavior" [Simon 67]. This link between emotions and the interruption of behavior had in fact been hanging around for some time in psychological literature [Angier 27] [Mandler 64], but had not found a proper grounding, perhaps for a lack of the concrete embodiment the computer metaphor could offer. Be it as it may, it is clear that the processes raised when the proper signals are triggered should receive almost the highest processing *PRIORITY*, to enable them to easily gain control, if the current threat or opportunity appears to be really critical. *This is why we proposed that emotional processes should operate as a control structure.* Actually the best comparison we have in mind so far is some mechanism akin to the *exception handling system* (EHS) designed in languages like Smalltalk-80 [Dony 90], although a proper embodiment would be likely to require that the specifications be realized as close as possible to the machine level. An *exception* is defined as any situation that "threatens" the current computation from achieving completion.

> "An EHS allows users to signal exceptions and to define handlers. To signal an exception amounts to (1) identify the exceptional situation, (2) to interrupt the usual sequence, (3) to look for a relevant handler and (4) to invoke it while passing it relevant information. Handlers are defined on (or attached to, or associated with) entities for one or several exceptions (according to the language, an entity may be a program, a process, a procedure, a statement, an expression, etc). Handlers are invoked when an exception is signaled during execution or the use of a protected entity. To handle means to set the system back to a coherent state..." (p. 323)

For such a system to operate as a regulator of commitments, another prerequesite is that all agents belonging to the same community have been built according to the same *ONTOLOGY*, so they can not escape from responding to the emotional acts emitted by other members. Indeed, the expression of anger only makes sense if the recipient is constrained (through some built-in design) to react to it significantly (otherwise, it would likely have already been dropped from the behavior repertoire, through natural selection!). Hence, one should envision, for every category of emotions, some built-in detector that would be triggered if the appropriate reaction is mandatory (for instance if there has been goal-blocking, threat, danger or loss...), and some list of preferred actions towards the target agent, from which the system should choose. Here again, there is much work from the psychology of emotions that suggests some universal antecedents for a small set of emotions. A more detailed account of the appraisal structure for basic emotions and their associated triggering signals is presented in [Aubé & Senteni 96].

For each emotion from an animat's repertoire then, the appraisal structure would specify the appropriate signals in terms of the commitments concerned (whether threatened or favored) and also provide an adequate *HANDLER* as a list of prefered actions, from which repair of the injured commitment would be carried out or any new opportunities exploited. Typical reactions for fear would involve fleeing, freezing, attacking, calling for help, broadcasting alarms... *One essential component of all handlers will have to deal with emotional expression,* the quasi-intentional structure of which we have compared elsewhere to the one specified for speech acts [Aubé & Senteni 96]. This communicative aspect [Oatley 92] has profound consequences for MAS theorizing in that it offers a key to the understanding of the emerging aspect of commitments envisioned by the social interactionists [Gerson 76] [Gasser 91]. The basic idea is that the expression of emotions does not merely act upon the inner mental states of the agents. *By the very same process, it also acts so as to distribute the problem (threat, danger, violation of standard, rupture of commitment...) as well as its computation over the whole community of agents, thus making for a very powerful tool of distributed control.*

Another important feature of our specifications is that all emotions be connected into a kind of *CONVERSATION NETWORK*, much in the spirit of the state transition diagrams for speech acts suggested by [Winograd & Flores 86] or the "interchange protocols" of [Campbell & d'Inverno 90]. The point here is that emotional reactions are themselves events that are designed to act upon commitments and should in turn trigger emotions in the others. Depending upon the specific configuration of commitments between two agents, fear could thus be answered with fear (alarm), with anger (so as to impose more fear), with sadness (as in parents being sorry for their child's fear), with guilt (if one has unwillingly scared the other) or with pride (if one has made the opponent retreat), but never with joy, hope or gratitude.

Last, but most importantly, we have to make provision for how *COMMITMENTS* themselves should be represented, so they could play the crucial role they bear within the whole architecture. One might think of a very simple numerical representation, since we advocated that they could be precisely measured in terms of the amount of first-order resources involved: how much has been allocated from others, or how much from self is going to be constrained. But clearly, this would not suffice, since they

would certainly have to register many more attributes, such as which acquaintance is concerned, or what is the time range for the commitment to be satisfied. Such requirements suggest that commitments could be envisioned as more active entities, operating concurrently, such as *actors* [Agha 86], that could themselves monitor their fluctuation with all the agents one is related to, and that could send messages to the appropriate emotional handlers whenever their value attribute is critically modified.

It also seems important to specify two special kinds of *LONG-TERM COMMITMENTS*. One is "affiliation" (alliance, attachment, friendship...) defined as some kind of contractual dependency between two agents, that could be increased or decreased depending on past history of helping and cooperation. The other is "power" (or status?), defined as socially recognized capacity for requesting help or resources from others, or for imposing consensus among subordinates. Those would be represented as kinds of "stored commitments" that could be invoked in certain situations (just as one could accumulate first-order resources to be used in periods of starvation). Except for the time scale, they share all the attributes of regular commitments. Those two seem quite useful for interpreting Bully's behavior presented in the beginning of the paper. It is fairly clear that the intrusion of Hirschmann in Bully's life was felt as considerable threat to the long-term commitment of affection between Lorenz and Bully, hence the frequency of the fights against the agent presumed responsible. But when Bully accidentally bit his beloved master, he himself betrayed at the very same time *two* strongly established long-term commitments - attachment and submission - which triggered a corrective handler of guilt with very high priority. The interruptive capacity of this one has to do with the strength of the affiliation, but also with the combined effect of having challenged the power commitment as well. Bully's guilt had to act upon Lorenz as an expression of submission and regret, but also upon Bully's internal states, in order that he would stamp firmly the situation in memory so as not to take the slightest risk of ever repeating it again.

5. Conclusion and future works

The nature of autonomous agents underlying animats' design is strongly conditioned by the hypothesis that emotional mechanisms are deeply wired in the repertoire of higher species. The central question addressed in the paper - "What are emotions for, in terms of benefits for the individual?" - is partially answered by the proposition of a coherent line of explanation of emotional behavior, stressing the question of adaptiveness. The first contention of the paper is that the underpinning of motivation provides a line of attack to this design problem. Autonomy and motivation are shown as intimately related: if agents are ever to get autonomous, it will only be through getting control over their motivations, by having ways of setting their own goals. The second claim of the paper is that, basically, motivations have to do with resource management. That is why behavioural ecology is so efficient in explaining animal behavior and why it would be very surprising that artificial autonomous agents could be designed otherwise. This concern leads to a closer study of the concept of resource that can be refined by distinguishing first-order and second-order. First-order resources are directly consumable ones that an agent already has at disposal. Second-order resources are those that an agent gets only indirectly, through having other agents committed to give them to him: the commitment is a resource in mediating access to some good. Being specifically defined in terms of the resources, commitments thus appear as one critical reason for setting up multi-agent societies: by regulating exchange, they enable the agents to use one another as a way to access additional resources that would otherwise remain out of reach, or would be too costly to obtain.

The third claim of the paper is that, within complex species that pursue multiple goals, commitments become the most important of all resources, one important corollary being that multi-agent societies require an effective control mechanism to regulate and control commitments fluctuations. Within animals, needs, such as those associated with states of hunger or thirst, are the motivational processes managing and regulating directly consumable first-order resources while, in higher-order species, emotions, such as anger or fear, are the motivational processes managing and regulating second-order resources or commitments. Like needs, emotions are powerful control structures protecting agents and societies of agents from running out of resources. As for needs, these control processes are triggered by significant variations in the flow of resources, for instance when there is a threat of breach in commitments or an opportunity for establishing new ones. An additional contribution is to propose one clear way of implementing the desired control structure through the use of a mechanism similar to the exception handling system of Smalltalk-80. Signals are defined within the appraisal structure of basic emotions, and the handlers involve communicative emotional acts plus a list of preferred actions or partial plans to be used for the repair of the injured commitment. One final claim stressed the theoretical utility of specifying special kinds of long-term commitments, such as affiliation and power.

Effective resource management in a distributed system requires a powerful control structure yet to be founded. Next generation parallel programs as well as next generation autonomous agents could benefit from a principled model of control, rooted in the psychology of human emotions, that would introduce emotion-like control structures. Such a mechanism would fit naturally into an inter-agent communication scheme and its introduction would likely improve the potential autonomy of agents by taking explicitly into account the management of the resources at the core of their activity. There is a lot of work remaining to be done along these lines, if we want to be able to express these control structures through readable and

reusable high level abstractions. Another thread of work would be to integrate the concerns expounded in this paper with other high-order concepts such as reflection. It is our contention that emotion-like control structures that serve to regulate resource consumption, could offer one indispensable complement to reflective structures that are also designed in to handle problematic situations. Metaphorically speaking, it is somewhat like adding thermodynamics to mechanics for a more complete account of the processes that are to be described.

References

[Agha 86] Gul Agha. *Actors*. Cambridge, MA: MIT Press, 1986.

[Angier 27] Roswell P. Angier. The Conflict Theory of Emotion. *American Journal of Psychology,* 39, 390-401.

[Aubé & Senteni 95] Michel Aubé and Alain Senteni. A Foundation for Commitments as Resource Management in Multi-Agents Systems. In Tim Finin and James Mayfield, editors, *Proceedings of the CIKM Workshop on Intelligent Information Agents*, Baltimore, Maryland, December 1-2 1995.

[Aubé & Senteni 96] Michel Aubé and Alain Senteni. Emotions as Commitments Operators: A Foundation for Control Structure in Multi-Agents Systems. In Walter Van de Velde and John W. Perram, editors, *Agents Breaking Away,* Proceedings of the 7th European Workshop on MAAMAW, Lecture Notes in Artificial Intelligence, No. 1038, p. 13-25, Berlin: Springer, 1996.

[Bond 90] Alan H. Bond. A Computational Model for Organizations of Cooperating Intelligent Agents. *SIGOIS Bulletin*, 11, 21-30, 1990.

[Brooks 91] Rodney A. Brooks. Intelligence Without Representation. *Artificial Intelligence*, 47, 139-159, 1991.

[Campbell & d'Inverno 90] John A. Campbell and Mark d'Inverno. Knowledge Interchange Protocols. In Yves Demazeau and Jean-Pierre Müller, editors, *Decentralized A.I.,* p. 63-80, North-Holland: Elsevier Science Publishers, 1990.

[Castelfranchi 90] Cristiano Castelfranchi. Social Power. A Point Missed in Multi-Agent, DAI and HCI. In Yves Demazeau and Jean-Pierre Müller, editors, *Decentralized A.I.,* p. 49-62, North-Holland: Elsevier Science Publishers, 1990.

[Castelfranchi 94] Cristiano Castelfranchi. Guarantees for Autonomy in Cognitive Agent Architecture. In Michael J. Wooldridge and Nicholas R. Jennings, editors, *Intelligent Agents,* Proceedings of the ECAI-94 Workshop on Agent Theories, Architectures, and Languages, Lecture Notes in Artificial Intelligence, No. 890, p. 56-70, Berlin: Springer-Verlag, 1994.

[Cohen & Levesque 90a] Philip R. Cohen and Hector. J. Levesque. Intention is Choice with Commitment. *Artificial Intelligence*, 42, 213-261, 1990.

[Cohen & Levesque 90b] Philip R. Cohen and Hector. J. Levesque. Rational Interaction as the Basis for Communication. In Philip R. Cohen, Jerry Morgan & Martha E. Pollack, editors, *Intentions in Communications,* p. 221-255, Cambridge: MIT Press, 1990.

[d'Inverno & Luck 96] Mark d'Inverno and Michael Luck. Formalizing the Contract Net as a Goal-Directed System. In Walter Van de Velde and John W. Perram, editors, *Agents Breaking Away,* Proceedings of the 7th European Workshop on MAAMAW, Lecture Notes in Artificial Intelligence, No. 1038, p. 72-85, Berlin: Springer, 1996.

[Dongha 94] Paul Dongha. Toward a Formal Model of Commitment for Resource Bounded Agents. In Michael J. Wooldridge and Nicholas R. Jennings, editors, *Intelligent Agents,* Proceedings of the ECAI-94 Workshop on Agent Theories, Architectures, and Languages, Lecture Notes in Artificial Intelligence, No. 890, p. 86-101. New York: Springer-Verlag, 1994.

[Dony 90] Christophe Dony. Exception Handling and Object-Oriented Programming: Towards a Synthesis. In Norman Meyrowitz, editor, *OOPSLA ECOOP'90 Proceedings, Ottawa, October 21-25 1990, Sigplan Notices*, 25 (10), 322-330, 1990.

[Ekman, Friesen & Simons 85] Paul Ekman, Wallace V. Friesen and Ronald C. Simons. Is the Startle Reaction an Emotion? *Journal of Personality and Social Psychology*, 49, 1416-1426, 1985.

[Ferber 96] Jacques Ferber. *Cooperation Strategies in Collective Intelligence.* Invited talk presented at the 7th European Workshop on MAAMAW, Eindhoven, The Netherlands, January 23 1996.

[Gasser 91] Les Gasser. Social Conceptions of Knowledge and Action: DAI Foundations and Open Systems Semantics. *Artificial Intelligence*, 47, 107-138, 1991.

[Gerson 76] Elihu M. Gerson. On "Quality Of Life". *American Sociological Review*, 41, 793-806, 1976.

[Hewitt 85] Carl Hewitt. The Challenge of Open Systems. *Byte,* 223-242, April 1985.

[Hewitt 91] Carl Hewitt. Open Information Systems Semantics for Distributed Artificial Intelligence. *Artificial Intelligence*, 47, 79-106, 1991.

[Jennings 93] Nicholas R. Jennings. Commitments and Conventions: The Foundation of Coordination in Multi-Agent Systems. *Knowledge Engineering Review,* 8, 223-250, 1993.

[Krebs & Davies 87] J. R. Krebs and N. B. Davies. *An Introduction to Behavioural Ecology.* Second Edition. Oxford: Blackwell Scientific Publications, 1987.

[Lorenz 70] Konrad Lorenz. *Tous les Chiens, Tous les Chats.* Paris: Flammarion, 1970.

[MacLean 93] Cerebral evolution of emotion. In Michael Lewis and Jeannette M. Haviland, editors, *Hanbook of Emotions,* p. 67-83. New York: The Guilford Press, 1993.

[Maes 91] Pattie Maes. Situated Agents Can Have Goals. In Pattie Maes, editor, *Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back,* p. 49-70. Cambridge, MA: MIT Press, 1991.

[Mandler 64] George Mandler. The interruption of behavior. In David Levine, editor, *Nebraska Symposium on Motivation,* p. 163-219. Lincoln: University of Nebraska Press, 1964.

[Oatley 92] Keith Oatley. *Best Laid Schemes: The Psychology of Emotions.* Cambridge, England: Cambridge University Press, 1992.

[Öhman 86] Arne Öhman. Face the Beast and Fear the Face: Animal and Social Fears as Prototypes for Evolutionary Analyses of Emotion, *Psychophysiology,* 23, 123-145, 1986.

[Reite & Field 85] Martin Reite and Tiffany Field, editors, *The Psychobiology of Attachment and Separation.* New York: Academic Press, 1985.

[Shoham 93] Yoav Shoham. Agent-Oriented Programming. *Artificial Intelligence,* 60, 51-92, 1993

[Shoham & Tennenholtz 95] Yoav Shoham and Moshe Tennenholtz. On Social Laws for Artificial Agent Societies: Off-Line Design. *Artificial Intelligence,* 73, 231-252, 1995.

[Simon 67] Herbert A. Simon. Motivational and Emotional Controls of Cognition. *Psychological Review,* 74, 29-39, 1967

[Toates 86] Frederick Toates. *Motivational Systems.* Cambridge, England: Cambridge University Press, 1986.

[Winograd & Flores 86] Terry Winograd and Fernando Flores. *Understanding Computers and Cognition,* New York: Addison Wesley, 1986.

Reinforcement Learning and Animat Emotions

Ian Wright
The Cognition and Affect Group
School of Computer Science
The University of Birmingham
Birmingham, B15 2TT, England
I.P.Wright@cs.bham.ac.uk

Abstract

Emotional states, such as happiness or sadness, pose particular problems for information processing theories of mind. Hedonic components of states, unlike cognitive components, lack representational content. Research within Artificial Life, in particular the investigation of adaptive agent architectures, provides insights into the dynamic relationship between motivation, the ability of control sub-states to gain access to limited processing resources, and prototype emotional states. Holland's learning classifier system provides a concrete example of this relationship, demonstrating simple 'emotion-like' states, much as a thermostat demonstrates simple 'belief-like' and 'desire-like' states.

This leads to the conclusion that valency, a particular form of pleasure or displeasure, is a self-monitored process of credit-assignment. The importance of the movement of a domain-independent representation of utility within adaptive architectures is stressed. Existing information processing theories of emotion can be enriched by a 'circulation of value' design hypothesis. Implications for the development of emotional animats are considered.

1 Introduction

Motivation and emotion are often conspicuously absent from information processing theories of mind. For example, Allen Newell's description of the SOAR architecture (Newell, 90; Laird et al., 87), the most advanced candidate for a unified theory of cognition, lists motivation and emotion as missing elements that need to be included in a more comprehensive theory.

Research into complete agent designs, such as 'animat' research within the field of Artificial Life (ALife), forces designers to attempt to integrate motivation, learning, sensing and acting within a single agent design to produce adaptive behaviour. If properly conceptualised, this work yields insights into the dynamic relationship between motivation, the ability of control sub-states to gain access to limited processing resources, and prototype emotional states. In particular, Holland's learning classifier system, a type of complex adaptive system often used in ALife research, provides a concrete, if simplified, example of this relationship.

A theoretical conclusion follows: the feeling component of some emotional states arises from the self-monitoring of a process of credit-assignment occurring within motivational subsystems. This conclusion enriches previous information processing theories of emotion and has implications for ALife research.

First, reasons are given why emotion may be thought 'difficult' for information processing theories.

2 Thinking refers but feelings just are

Cognitive representations can denote or refer to states-of-affairs that exist in an agent's domain. This is the physical symbol system hypothesis – the hypothesis that physical systems can implement symbols that contain information that denotes (Simon, 95). For example, an animat within a simulated domain may possess information about other agents in the environment, including their type, location, or speed. Such information sub-states of the animat are causally linked to their referents: the representation of the speed of agent A will alter if it is perceived that agent A has altered its speed. This is a simple example: referential links can be very indirect in more complex information processing systems. The principle, however, is conceptually clear and forms a basis of information processing theories of mind: *thinking refers.*

The emotions, however, differ from 'cold' cognition: they can be 'hot', often involving feelings of pleasure or displeasure with associated intensities. Unlike 'straightforward' representational thinking, an emotional state has *both* a representational content, e.g., a state of happiness *about* passing one's exams, and a hedonic, or *valenced* content, e.g., the particular form of intense pleasure one is experiencing. The hedonic component does

not represent a state-of-affairs: *feelings just 'are'.*

Many information processing theories of emotion tend to avoid an explanation of the hedonic components of emotional states by concentrating on the semantics of representational components (e.g., Dyer's BORIS system (Dyer, 87), Frijda and Swagerman's ACRES system (Frijda & Swagerman, 87), and Pfeifer's FEELER system, reviewed in Pfeifer, 94). Alternatively, feelings are brushed under the physiological carpet by assuming that all valenced states arise from perceptions of bodily states. For example, Herbert Simon in his seminal paper on motivation and emotion (Simon, 67), outlines a view of 'feelings' that closely resembles William James' peripheric theory of the emotions: '... sudden intense stimuli often produce large effects on the autonomic nervous system, commonly of an "arousal" and "energy marshaling" nature. It is to these effects that the label "emotion" is generally attached'; and '... the feelings reported are produced, in turn, by internal stimuli resulting from the arousal of the autonomic system'. It is difficult to conceive how this view of valency could account for the mental pain associated with, for example, grief, which does not necessarily require bodily arousal or disturbance.

Therefore, there appear to be at least two reasons why explanations of hedonic states are generally absent from cognitive theories: first, valenced components appear not to conform to the representational model that supports cognition; and second, their possible functional role is unclear: what can these states possibly *do* if they do not represent? Why do such diverse and complex states such as happiness, sadness, glee, triumph, grief, despair, intense disappointment etc. have hedonic components?

2.1 A preliminary definition of valency

Such 'states' are phenomenologically highly variegated (compare your memories of being angry with being happy), with different causal antecedents (e.g., being slighted before your peers, or winning an Olympic gold medal) and different consequences (such as a desire for revenge or rest). The folk-psychological concepts that refer to these states play a communicative role between agents, yet ultimately derive from sophisticated internal self-monitoring mechanisms, a kind of 'internal perception' (Wright, Sloman & Beaudoin, to appear). This ability gives rise to the method of introspection or phenomenological analysis in psychology. Much theoretical work in the emotions draws on introspection, and this kind of knowledge places important constraints on possible theories. However, there is no reason to believe that the knowledge gained from our internal perception is any less fallible than that gained from 'external' perception. In the absence of good theories of the underlying mechanisms, care is required when employing phenomenological concepts.

A preliminary division can be made between *dispositional* and *occurrent* emotional states (Ryle, 49; Green, 92). A dispositional state is a latent state that may manifest in appropriate circumstances, such as the brittleness of a wine glass, whereas an occurrent state is a *running* state, such as the process of a wine glass breaking. For example, a man who has lost a parent may function normally at work (dispositional grief), only to break down in the evening (dispositional state manifests as an occurrent emotion).

There are two distinguishable components of an occurrent emotional state: *intentional* and *non-intentional.* The intentional component of an emotional state is what the state is about. A person is angry, disappointed, or ecstatic 'about' a perceived state of affairs[1]. This state of affairs may exist in the agent's environment, or entirely within cognition (as in the case of the mathematician irritated with himself for being unable to solve an equation). The intentional component has representational content.

The non-intentional component of an emotional state is often referred to as its 'hedonic tone', *feeling,* or *valency.* 'Feeling' is an ill-defined word, for it can cover such diverse sensations as one's cheeks burning with embarrassment, an itch on the left ear, or the mental happiness associated with triumph. 'Hedonic tone' is similarly semantically overloaded: it can be used to refer to the enjoyable sensation of a full stomach after a large and hearty meal. The word valency, if given a suitable definition, can avoid such confusion. Before giving such a definition we require some more distinctions.

A division can be made between *physiological* forms of pleasure and displeasure, and *cognitive* valency. For example, the (self-monitored) 'itchiness' on my left ear is a form of displeasure linked to information concerning bodily location. In contrast, the (self-monitored) mental pain of intense grief is a form of displeasure linked to information about a loved one's death. There can be no 'pain receptors' for this kind of displeasure, unlike the nerves that detect a pin pricking one's finger.

To illustrate: an athlete may be experiencing the occurrent emotional state of triumph while standing on the winner's podium. The intentional component of her state includes thoughts pertaining to her achieved goals; the non-intentional component *includes* the feeling of increased heart-rate or arousal, the warm sun beating on her brow, *and* a valenced state of cognitive pleasure not located on or in the body.

'Happiness' and 'sadness' provide the clearest examples of valency. The discussion, therefore, will restrict itself to these emotional states. Particular examples of happiness are triumph, glee, joy, ecstasy, gladness, love; and of sadness, despair, disappointment, grief, and sorrow. In a similar way to object-oriented programming languages, a preliminary taxonomy can be constructed in

[1] Moods can be considered an exception.

which 'happiness' and 'sadness' define classes of emotions with *valence* and *intention* slots. Particular instances of these generic emotions have additional slots that define finer-grained attributes. In this way, a hierarchy of subtypes of the emotions 'happiness' and 'sadness' is formed (see Ortony, Clore & Collins, 88 for a similar treatment). Consequently, the terms 'happiness' and 'sadness', in this paper, refer to generic, abstract definitions. For the sake of brevity, it will be stated that a person is 'happy' when a desired goal is achieved, and 'sad' when failure occurs in achieving a desired goal. This is an oversimplification: concrete emotional states are rarely this straightforward or as simple.

Unlike the intentional component of an emotional state, valency does *not* represent a state of affairs. It can differ qualitatively in only a very restricted sense, i.e. it can be *either* pleasurable or displeasurable, and allow quantitative degrees of *intensity*: the valency can be very displeasurable or only mildly so. From a phenomenological perspective, valency is a 'brute fact'[2] of one's present state, unlike beliefs that can be true or false, or goals that have been achieved or not.

A preliminary definition of valency can now be provided: *Valency* is a form of pleasure or displeasure not located in the body, and is *a* non-intentional component of occurrent emotional states of happiness or sadness.

Our problem is to provide a theoretical account of valency.

3 A design-based answer: thermostats and classifiers

There are almost as many theories of emotion as emotion theorists. The field, as has often been remarked, is characterised by terminological confusion (Kagan, 78; Read & Sloman, 93) and riven by differing 'schools of thought' (Pfeifer, 94).

Approaches to the study of emotions can be very broadly categorised as *semantics*-based, *phenomena*-based and *design*-based (Sloman, 92). Semantics-based theories analyse the use of language to uncover implicit assumptions underlying emotion words (e.g., Wierzbicka, 92). Phenomena-based theories assume that emotions are a well-specified category and attempt to correlate contemporaneous and measurable phenomena with the occurrence of an emotion, such as physiological changes (an early example is William James' theory – see Calhoun & Solomon, 84; for a comprehensive review of many phenomena-based theories, see Strongman, 87).

The design-based approach, in contrast, takes the stance of an *engineer* attempting to build a system that exhibits the phenomena to be explained, and is a 'rational reconstruction' of the practice of Artificial Intelligence, considered as the general science of intelligent systems, both natural and artificial (Sloman, 93b). The methodology involves exploring an abstract space of possible requirements for functioning agents (*niche-space*) and the space of possible designs for such agents (*design-space*) and the mappings between them (Sloman, 95). This is an iterative process and the research strategies can vary: they may be top-down, bottom-up or 'middle-out'. All are potentially useful.

Research within the field of Artificial Life is an example of the design-based approach, and is characterised by the investigation of complete agents that integrate many capabilities, such as sensing, action selection, acting and (particularly) adaptation to a simulated or real niche. A consequence of such a methodology is that the analytic isolation of emotions from other cognitive phenomena, often characteristic of semantic and phenomena-based approaches, can be directly avoided by the investigation of complete systems. This methodological course is likely to bear the most fruit and have the most relevance for unified theories of cognition.

The development of negative feedback control systems demonstrated that simple, materially embodied systems can have sub-states with different functional roles, in particular 'belief-like' and 'desire-like' control sub-states (McCarthy, 79; Sloman, 93a; Powers, 88; Braitenburg, 84). For example, the belief-like sub-state of a thermostat is the curvature of its bi-metallic strip, which alters in accordance with the ambient temperature of a room; the desire-like sub-state is the setting of the control knob. Negative feedback ensures that the temperature of the room stabilises around the control knob setting, i.e. the thermostat 'acts' in the world to achieve its 'desire'. Of course, a thermostat does not have sufficient architectural complexity required to support human beliefs and desires, but it is an illustrative 'limiting case' (Sloman, 93a).

The simple thermostat implements a function that maps an input temperature to an output signal, which controls a heater. It does not learn. We can move through design-space adding architectural complexity to the negative feedback loop, such as varying the kinds of sub-states, the number and variety of sub-states, the functional differentiation of sub-states, and the kinds of causal influences on sub-states, such as whether the machine can change its own desire-like states, and so on (see Sloman, 93a for an extended discussion).

Holland's classifier system (Holland, 95; Holland, 75; Holland et al., 86; Riolo, 88) is one of a class of relatively well-understood machine learning algorithms. Unlike the thermostat, it is sufficiently complex to exhibit prototypes of 'emotion-like' states, much as the thermostat exhibits prototypes of 'belief-like' and 'desire-like' sub-states. An analysis of its functioning reveals an important role for non-intentional representations. First, a brief description of a classifier system.

[2]This term borrowed from (Chalmers, 96).

3.1 The learning classifier system

A classifier system consists of a *performance system*, and *credit-assignment* and *rule discovery* algorithms.[3] The performance system consists of a *classifier list* that consists of a set of condition-action rules called *classifiers*, a *message list* that holds current messages (as per the working memory of production systems), an *input interface* that provides the classifier system with information about its environment in the required form, and an *output interface* that translates action messages into world events. The *basic cycle* of the performance system matches messages in the message list (including sensory messages) with classifiers, which then post their actions 'back' to the message list. Many classifiers may become active and fire in parallel. Any current action messages are sent to the output interface. The performance system is a universal machine; that is, any computable function can be implemented as a collection of classifiers.

The performance system alone cannot learn. The credit-assignment algorithm in the classifier system is a *bucket-brigade* (*bb*) algorithm. This algorithm introduces competition between classifiers based on a quantitative 'strength'. Each classifier that has its condition activated by a message *bids* to post its action part to the message list. Only the highest bidders are allowed to post their actions. The bid of a classifier depends on its strength[4], which is a measure of the classifier's 'usefulness' to the system. The higher the strength of a classifier the more likely it will win the competitive bidding round and post a message. The behaviour of the classifier system, therefore, can be modified by changing the strengths associated with individual classifiers. If the strength of the classifiers that tend to lead to 'useful' behaviour can be increased, and the strength of the classifiers that tend to lead to 'useless' behaviour can be decreased, the system will learn to produce more useful behaviour. The *bb* is designed to bring about these types of changes in strength.

The basis for the *bb* is information from *reinforcement mechanisms* about whether the classifier system as a whole is behaving correctly. This is achieved via *rewards*, i.e. the system will receive positive reward when it behaves correctly and negative reward when it behaves incorrectly. More often than not neither positive or negative reward will be received. This is a type of *reinforcement learning* (for a survey, see Kaelbling et al., 95), which derives its name from behaviourist theories of animal learning (e.g., Mackintosh, 83). The phrase 'credit assignment' will be preferred over 'reinforcement learning': the former emphasises internal mechanisms, whereas the latter is often associated with external operations.

When a reward is received the *bb* adds the reward value to the strength of all classifiers currently active, thereby changing the strength of classifiers *directly* associated in time with useful behaviour. Also, when a classifier is activated it 'pays' the amount it bid to the antecedent classifier that produced the message it matched. The strength of the active classifier is decreased by its bid amount. In this way, the *bb* acts to increase the strength of classifiers *indirectly* involved in the production of useful behaviour. This is a partial solution to the *temporal credit assignment problem*, which is the problem of determining which antecedent rules to strengthen given that there can be long delays between antecedent classifiers and the resultant rewarding act. The *bb* allows reward to 'circulate back' through the system (in a similar way to the 'back-propagation' of an error signal in artificial neural networks). Chains of high strength classifiers, performing useful computation, can emerge from such a scheme.

An alternative name for the quantitative measure associated with each individual classifier is *value*. This switch in terminology can be better understood by recalling that the *bb* was originally inspired by an economic metaphor (Holland et al., 86), in which classifiers are agents (consuming and producing messages) who possess a certain amount of money ('strength' or value) which they exchange for commodities at the market (the message list or blackboard). Much as money mirrors the flow of commodities in a simple commodity economy, *value mirrors the flow of messages* in the learning classifier system.

The *bb* can lead to improved system behaviour through the selection of some classifiers over others; however, it cannot create entirely new classifiers. A genetic algorithm (*ga*) implements rule discovery. Periodically, classifiers of high strength are selected as parents. The genetic operators of *crossover* and *mutation* are applied to classifiers considered as chromosome strings. The resultant offspring are placed in the classifier list. Normally, the *ga* is applied less frequently than the *bb* otherwise new classifiers will not have had sufficient time to be evaluated.

To summarise: within a classifier system *selection on value* occurs twice: in bidding rounds of classifiers competing for messages, and in rule discovery where high value classifiers generate offspring and low value classifiers are eventually removed from the system altogether. The combined effects of *circulation of value* via the *bb* and a double selection on value by competition and rule discovery allows the classifier system to reallocate classifier rules from unrewarding to rewarding processing. This ability makes it an adaptive system.

The classifier system, when embedded in a simulated

[3] This specification abridged from (Riolo, 88). Readers unfamiliar with classifier systems should consult (Holland, 95).

[4] This is a simplification for the sake of brevity. The bid of a classifier can depend on at least three factors: strength, the 'specificity' of the classifier, which is a measure of its relevance to a particular set of messages, and 'support', which allows internal messages to have differential importance.

or real environment, can construct classifiers that produce satisficing behaviour given the constraints and guidance of reinforcement mechanisms. It has been extensively used in the ALife community (Steels, 94), for example in developing control programs for robots through supervised learning (Dorigo & Colombetti, 93).

3.2 The intentional component of classifier states

The internal state of an implemented and *running* classifier system is continually changing. We will denote the classifier system's internal state at time step t as C_t, and define it as the joint operation of the basic cycle and *bb* considered as one indivisible moment. C_t has two distinguishable components: *intentional* and *non-intentional.*

The intentional component of C_t is the message list containing messages with representational content. The simplest example of the representational content of messages can be found in (Holland, 95). We can imagine an artificial frog embedded in an environment of real or simulated flies. The control program for *simfrog* is a classifier system with an adapted set of classifiers. The *simfrog* has an eye sensor, which forms part of the classifier system's input interface. The eye can detect a number of attributes of any fly within range. Attributes could include whether the fly is moving, what colour it is, its size and proximity. If a fly is detected the eye sensor posts a message to the message list that encodes this information. This sensory message is the result of a *mapping* between a state of the environment and a sub-state of *simfrog.* It is in this sense that messages have representational content. Internal messages, less directly linked to sensing or acting, will have more complex representational roles within the system. The *semantics* of messages depends on the dynamic relationship between message and environment. For example, the sensory message may match a classifier that posts an action message that results in *simfrog* throwing its sticky tongue in the direction of the detected fly. The meaning of the message, therefore, would be an impoverished version of 'eat that fly!'.

3.3 The non-intentional component of classifier states

The non-intentional component of C_t is the circulation of value between antecedent classifiers, messages and matching classifiers. Value is exchanged for messages, i.e. matching classifiers pay the 'owners' of messages an amount of value, via the *bb*. Unlike messages, the value that circulates has no representational content. The values associated with classifiers specify a probabilistic partial ordering on the classifier list. The ordering is partial because only some classifiers will bid for the same message. The ordering is probabilistic because the classifier selection mechanism is stochastic.

For example, both classifiers c_i and c_j match message m; c_i has value v_i, and c_j has value v_j, with $v_i > v_j$. In competitive bidding for message m, c_i will out bid c_j more often than not, i.e. there is a probabilistic total ordering on the set $A = \{c_i, c_j\}$. Other classifiers in the classifier list, such as c_k, *never* compete against any $c \in A$; consequently, no ordering holds between them[5]. Classifier values, therefore, specify a partial and probabilistic relation of utility over the classifier list. Value is internal economy alone. It does not represent anything within or external to the classifier system; rather, it specifies a *relation* between classifiers. It is this property of value that helps to make the classifier system a *domain-independent* learning algorithm: the representation of utility does not alter from domain to domain.

The classifier system tightly integrates impoverished conceptions of cognition, conation and affect.

The cognitive engine is the performance system consisting of condition-action rules and a global blackboard (cf. the production system and working memory of Newell's SOAR architecture).

Conation, or 'motivational force', is also represented in the classifier system. The value of a classifier is its dispositional and relative ability to fire and post a message. Whether the implementation of the classifier system is truly parallel (with perhaps separate processors for each classifier), or only simulates parallelism, *the value of a classifier is an ability to buy processing power.* A high value classifier will be more likely to win bidding rounds, be processed, and post its action part. An internal sensory message, for example a message that *simfrog*'s energy is below a danger threshold, may match the first in a chain of high value classifiers that instigate a search for flies in the environment. The high value of such a processing chain will make it unlikely that other rules will out bid and switch processing to other ends. In this impoverished sense, value is motivational force.

Finally, the operation of the bucket-brigade, which alters the 'buying power' of sets of classifiers, involves losses or gains of value that are ultimately derived from reinforcement mechanisms, i.e. information regarding the 'goodness' or 'badness' of agent-environment situations. Positive and negative reinforcement are often linked to pleasure and pain.

3.4 Self-monitoring of credit-assignment

We now add a simple 'self-monitoring' mechanism to the classifier system. The mechanism is required to monitor the circulation of value and send its output to a suitable device. For current purposes self-monitoring need play no functional role within the classifier system, so the de-

[5] In classifier systems with maximum size message lists the situation can become more complex, as classifiers compete for space in addition to competing for messages.

vice can be a computer screen that displays results to an ALife engineer.

C_t involves a *net* exchange of value, denoted V_t, from matching classifiers to antecedent classifiers. The self-monitoring mechanism records each V_t over a specified time period, say $t = 1 \ldots n$, and displays the *change* in value, denoted δV_t, which is exchanged from one time step to the next, where $\delta V_t = V_{t+1} - V_t$. δV_t can be either:

- Positive, implying (a) a net increase in the utility of antecedent classifiers, and (b) currently active classifiers are likely to lead to positively rewarding consequences;
- Negative, implying (a) a net decrease in the utility of antecedent classifiers, and (b) currently active classifiers are likely to lead to negatively rewarding consequences; or
- Zero, implying no net change in the utility of antecedent classifiers.

Therefore, the self-monitoring of $C_1 \ldots C_n$ will display a rate of change of value with both sign and magnitude. We now connect the output of self-monitoring to *simfrog*'s skin, which can change colour. If δV_t is zero *simfrog* remains *green*, if δV_t is positive he displays *yellow* with an intensity $|\delta V_t|$, and if δV_t is negative he displays *blue* with intensity $|\delta V_t|$. When *simfrog* catches and eats a fly he will blush bright yellow as innate reinforcement mechanisms strongly positively reward antecedent classifiers. If *simfrog* possessed more sophisticated reflective capabilities he might wonder why he has beliefs that refer *and* an odd quantitative intensity that is either positive or negative but doesn't seem to be 'about' anything or serve any apparent purpose. Depending on philosophical prejudice, one might be tempted to say that *simfrog feels* happy, sad or indifferent depending on circumstance.

Some caveats are in order. A classifier system is limited in many ways. It does not have an explicit memory store. It tends to be an entirely reactive system with no representation of goals. It does not anticipate, or perform prior search within a world model before acting. In real-world applications it can be difficult for a classifier system to learn appropriate behaviours (Wilson & Goldberg, 89). Also, the classifier system is not a fixed architecture but continues to evolve, e.g. the recent introduction of rule discovery based on the accuracy of classifier predictions (Wilson, 95). However, by abstracting from implementation details we can examine certain *design principles* embodied in the classifier system. The following section examines the implications of such design principles for information processing theories of human emotions.

4 Circulation of value

There are two kinds of finite computational[6] resource in the classifier system: (a) the total *information* capacity of the performance system, which consists of the set of 'if-then' rules, a (usually fixed) number of classifiers that perform simple computations, and (b) *processing* limits, which is the amount of parallel computation allowed per time step; for example, a maximum of ten classifiers may be allowed to fire during each basic cycle. For current purposes, information limits will be placed to one side.

4.1 Processing limits: emotions as interrupts

Existing information processing theories of emotion agree on the importance of *processing* resource limits in accounting for emotional states. For example, Simon's 'interrupt theory' associates emotional states with the interruption of a resource-bound, high level, attentive system due to new, perhaps urgent motives: 'The theory explains how a basically serial information processor endowed with multiple needs behaves adaptively and survives in an environment that presents unpredictable threats and opportunities. The explanation is built on two central mechanisms: 1. A goal-terminating mechanism [goal executor] ... [and] 2. An interruption mechanism, that is, emotion allows the processor to respond to urgent needs in real time.' (Simon, 67; Simon, 82). An interrupting stimulus, such as the presence of a predator, disrupts ongoing goal processing in the serial processor and substitutes new goals to deal with the new situation producing, amongst other things, emotional behaviour, e.g. the flight-fight-fright response.

Aaron Sloman's attention filter penetration theory (Sloman & Croucher, 81; Sloman, 87; Sloman, 92) extends the interrupt theory by introducing new, architectural detail implicit in Simon's paper. Two types of processing are distinguished: pre-attentive, highly parallel and automatic motive generation processes, and attentive, resource-bound 'motive management' processes that exhibit a limited degree of parallelism. The concept of *insistence* is introduced, which is the dispositional ability of a motivator to 'surface' through a variable threshold filter and disrupt attentive processing. The *intensity* of a motivator is its ability to 'keep hold' of attention once surfaced. A common characteristic of many emotional states is the phenomenon of *perturbance* (Beaudoin, 94), which occurs when a motive has been postponed or rejected but nevertheless keeps resurfacing to disrupt ongoing, motive processing. The concept of perturbance has been extensively used to provide an architectural account of grief or 'loss' (Wright, Sloman & Beaudoin, to appear). As in Simon's theory, 'emotional'

[6] A classifier system animat embedded in an environment will also have 'physical' resource constraints, such as the number and kind of effectors and detectors.

interrupt mechanisms are needed to maintain reactivity to important events in a system with finite processing resources.

Both interrupt theories rely (implicitly in Simon's case) on a distinction between *control* and *semantic* signals in information processing architectures. Semantic signalling is the propagation of information that has representational content, whereas control signalling does not refer or have semantic content but performs a control function, such as changing the control flow of the system, or putting it into a distinct kind of processing state[7]. It is the non-representational nature of control signalling that is held to account for the non-intentional, 'feeling' component of emotional states. However, a theoretical explanation of valency is absent: reasons why some control signals are pleasurable or displeasurable, i.e. possess a qualitative dimension of valency that can be either positive or negative, and also vary in quantitative intensity, are not provided.

Keith Oatley and Philip Johnson-Laird's (Oatley, 92; Oatley & Johnson-Laird, 85; Johnson-Laird, 88) communicative theory addresses this question by introducing basic, irreducible and phylogenetically older architectural control signals. 'Each goal and plan has a monitoring mechanism that evaluates events relevant to it. When a substantial change of probability occurs of achieving an important goal or subgoal, the monitoring mechanism broadcasts to the whole cognitive system a signal that can set it into readiness to respond to this change. Humans experience these signals and the states of readiness they induce as emotions' (Oatley, 92). Control signals, of which 'happiness' and 'sadness' are examples, communicate significant junctures of plans to other cognitive subsystems. Emotional states, therefore, are viewed as a design solution to certain problems of the transition between plans in systems with multiple goals. The functional role of valenced signals is to enforce state transitions by interrupting a central processing system; for example, the 'sadness' control signal, broadcast when a major plan fails, causes a state transition to search for a new plan. On this view, control signals differ in valency because they differ in their functional roles.

All three theories lack a consideration of *adaptive* architectures, i.e. architectures able to modify themselves to improve their behaviour. Explicitly considering adaptation provides a new functional role for control signalling.

[7]The following analogy may help capture the distinction. Imagine trains travelling on a complex network of tracks. Postal trains contain mail (semantic content) with destination addresses on the envelopes. These trains travel to the destinations and deposit the mail (the information). However, a different kind of train, a 'control signal' train, can travel through the network altering the points of the tracks. This has the effect of changing the topology of the network, i.e. trains will continue to deposit their mail but will use different routes.

4.2 Adaptation: changes in the ability to buy processing power

The phrase *circulation of value* denotes a general design principle, and refers to any mechanism that (i) alters the dispositional ability of agents in a multi-agent system to gain access to limited processing resources, via (ii) exchanges of a quantitative, domain-independent representation of utility that mirrors the flow of agent products[8]. A particular example of circulation of value is the bucket-brigade algorithm, where an 'agent' is a single classifier, 'multi-agent system' is the set of competing and cooperating classifiers[9], and 'agent products' are messages.

The circulation of value is a pattern of flow of control signals. Such signals have no semantic content and propagate around the system altering control flow. Additionally, classifier bids attempt to 'grab' processing resources and may 'interrupt' current processing causing internal or external behaviour to take different routes. A quantitative representation of utility, therefore, need not contradict the interrupt function identified by Simon and Sloman, nor the control signal function identified by Oatley and Johnson-Laird. The local exchange of value between classifiers can generate the negative and positive control signals of the communicative theory: instead of two signals, we now have one (see section 3.4). This is a more parsimonious state of affairs. But, additionally, there is a new, previously unidentified functional role for the control signal: the circulation of value implements a type of *adaptation*. This is not inductive learning of new hypotheses about a domain, but an ordering and reordering of the utility of control sub-states to dispositionally determine behaviour. In a classifier system, the circulation of value adaptively changes the ability of classifiers to buy processing power.

The design principle of circulation of value opens up the possibility of architectures that generate valenced states far removed from physiology. High level cognitive processes may be saturated with value, allowing the production of semantic messages coupled with losses or gains in quantitative value. For example, negatively valenced states, such as grief, may, amongst other things, involve

[8]This definition is very general; for example, it applies to currency flow in simple commodity economies (limited processing resources would correspond to the available labour-power of the social system). However, *circulation* of value does not usefully apply to tabular reinforcement learning algorithms, such as Q-learning, which use a quantitative representation of value without internal exchanges of value. An XCS classifier system (Wilson, 95) that allows chains of classifiers to form will exhibit circulation of value; however, the distinction between strength and accuracy of pay-off prediction may afford more sophisticated self-monitoring; in particular, the difference between predicted and actual pay-off may help introduce *expectancies* into the discussion. Such an analysis is beyond the scope of this paper.

[9]To be precise, a set of classifiers is not a multi-agent system, as the definition of an agent normally includes a requirement for beliefs and desires, whereas a classifier is only a production rule.

the gradual loss of the accumulated value of a structure of attachment (Bowlby, 79; Bowlby, 88; Wright et al., to appear) towards a loved one. Negative valency may be a necessary consequence of adaptive change: the structure of attachment, no longer useful for motive generation, loses its ability to buy processing power, grab attentive resources and dispositionally determine behaviour. This process is self-monitored as displeasurable.

These considerations lead to a *design hypothesis* of a quantitative representation of utility that circulates within a subset of cognition. The next step is the construction of a circulation of value theory of emotions that builds on previous work. This will require another iteration of the design-based approach. The design-space of autonomous agent architectures that integrate circulation of value learning mechanisms with more complex forms of motive management needs to be explored. The capabilities, properties and explanatory power of such designs can then be examined. Further comparisons can then be made between designs and psychological phenomena. Preliminary efforts in this direction are reported in (Wright, 95). In addition, it will be necessary to investigate the results and theories of neuropsychology in order to map the postulated mechanisms onto the neural substrate. If some form of circulation of value is found to occur in human brains this would add credence to such a theory. However, if the further exploration of adaptive architectures finds no use for circulation of value mechanisms then the hypothesis will need to be modified or replaced: ultimately, it is an empirical question.

However, even at this early stage it is possible to make some theoretical claims. Given the above considerations, and our preliminary definition of valency, we can state a corresponding architectural process that gives rise to it: *valency is a self-monitored process of credit-assignment.* The hedonic, or 'feeling' component of many emotional states arises from architectures that employ a circulation of value mechanism. (A type of) 'feeling' is (a type of) adaptation, and 'hot' cognition – forms of pleasure or displeasure involved in short term and long term control – need pose no insurmountable problems for information processing theories of emotion.

5 Towards emotional animats

The emotions encompass a broad range of human experience, whereas valency is a component of only some emotional states. To approach a real understanding of human emotions will require an investigation of more sophisticated architectures satisfying more complex requirements. This is long-term research. Yet it can be said with confidence that simple animats already exhibit simple 'emotion-like' states, as long as we take care over definitions and avoid hyperbole. 'Broad but shallow' agent designs can be illuminating. For example, (Balkenius, 95) provides a clear discussion on relations between simple motivations and simple emotions based on experiments with animats. Also, the simulation of societies of competing and cooperating adaptive agents will impose new requirements on architectures and give rise to new kinds of internal state. (Aube & Senteni, 95) view (more complex) emotional states as 'commitment operators' that manage resources in multi-agent systems.

An important difference between such design-based (e.g., see Beaudoin & Sloman, 93; Beaudoin, 94; Sloman, Beaudoin & Wright, 94; Pfeifer, 94) approaches and previous computer simulations of emotions is that they move from requirements for *complete agents* to possible designs, and do not directly 'program in' correlates of emotions. In this way, design features are linked to niche features providing explanations of *why* emotional states are present in nature. A small example is provided in this paper: a requirement for adaptability entails, at some level, credit-assignment mechanisms that use a domain-independent representation of utility or value, a kind of internal 'common currency'. Such a representation does not refer but is relational, and can be gained or lost depending on whether actions are successful or unsuccessful in leading to rewarding consequences. These kinds of processes are self-monitored as non-intentional states, or 'feelings', which are quantitative in nature and either pleasurable or displeasurable.

Those philosophically inclined may doubt that animats constructed in this way 'really' experience their non-intentional states. Is it *anything to be like* a simulated animat? This kind of question is experimentally undecidable, and has no engineering consequences whatever. It is therefore scientifically uninteresting.

6 Conclusion

A subset of emotional states was examined and the concept of valency – the non-intentional component of occurrent emotional states of happiness or sadness – was introduced. Valency is a preliminary definition of a subset of the states colloquially referred to as 'feelings'. The internal state of a complex adaptive system, the learning classifier system, able to function as a complete agent within a niche, was examined. Much like the simple thermostat exhibits prototypes of 'belief-like' and 'desire-like' sub-states, the classifier system was found to exhibit simple examples of valenced states.

The design principle of circulation of value was abstracted from the classifier system and employed to overcome existing inadequacies in information processing theories of emotion. The circulation of value mechanism is more parsimonious than Oatley and Johnson-Laird's control signalling, and, more importantly, it introduces a basic form of adaptation. These new considerations build upon and do not contradict previous theories. A theoretical conclusion is that, to a first approximation,

valency is a self-monitored process of credit-assignment. 'Feeling' is the self-monitoring of adaptation; that is, the non-intentional component of generic 'happiness' and 'sadness' states includes a movement of internal value, which functions to alter the dispositional ability of control sub-states to buy processing power and determine behaviour. Such a process is self-monitored as a 'brute' feeling because value does not refer, unlike beliefs that can be true or false, or goals that can be achieved or not.

To approach the complexity of concrete, human emotional states will require many more iterations of the design-based approach.

Acknowledgments

My thanks to Aaron Sloman, Chris Complin, Tim Kovacs, members of the Cognition and Affect and EEBIC (Evolutionary and Emergent Behaviour Intelligence and Computation) research groups at Birmingham, and the SAB96 referees for their helpful comments.

References

Aube, M., & Senteni, A. (1995) Emotions as commitments operators: a foundation for control structure in multi-agent systems. In *Proceedings of the 7th European Workshop on Modelling Autonomous Agents in a Multi-Agents World, MAAMAW'96,* Lecture Notes on Artificial Intelligence, No. 1038, Springer–Verlag.

Balkenius, C. (1995) *Natural intelligence in artificial creatures.* Lund University Cognitive Studies 37.

Bates, J., Loyall, A. B., & Reilly, W. S. (1991) Broad agents. Paper presented at the *AAAI spring symposium on integrated intelligent architectures.* Stanford, CA: (Available in SIGART BULLETIN, 2(4), Aug. 1991, 38-40).

Beaudoin, L. P. (1994) *Goal Processing in Autonomous Agents.* PhD Thesis, School of Computer Science, University of Birmingham.

Beaudoin, L. P. & Sloman, A. (1993) A study of motive processing and attention. In *Proceedings of AISB93,* A. Sloman, D. Hogg, G. Humphreys, A. Ramsay & D. Partridge (Eds), 229-238, Oxford: IOS Press.

Bowlby, J. (1979) *The making and breaking of affectional bonds.* Tavistock Publications Ltd.

Bowlby, J, (1988) *A secure base.* Routledge.

Braitenburg, V. (1984) *Vehicles – experiments in synthetic psychology.* MIT press.

Calhoun, C., & Solomon, R. C. (Editors) (1984) *What is an Emotion? Classical Readings in Philosophical Psychology.* Oxford University Press.

Chalmers, D. (1996) *The Conscious Mind.* Oxford University Press.

Dorigo, M. & Colombetti, M. (1993) Robot shaping: developing situated agents through learning. Technical report TR-92-040 (revised), International Computer Science Institute, Berkeley, CA.

Dyer, M. G. (1987) Emotions and their computations: Three computer models. *Cognition and Emotion* 1(3), 323-347.

Johnson-Laird, P. N. (1988) *The Computer and the Mind: An Introduction to Cognitive Science,* Fontana.

Kagan, J. (1978) On emotion and its development: a working paper. In *The Development of Affect* edited by M. Lewis and L. A. Rosenblum. Plenum Press, New York and London.

Frijda, N. H., & Swagerman, J. (1987) Can computers feel? Theory and design of an emotional system. *Cognition and Emotion,* 1, 235-257.

Green, O. H. (1992) *The Emotions,* a philosophical theory. Kluwer Academic Publishers.

Holland, J. H. (1975) *Adaption in natural and artificial systems.* MIT Press.

Holland, J. H. (1995) *Hidden Order, How Adaptation Builds Complexity.* Helix Books.

Holland, J. H., Holyoak, K. J., Nisbett, R. E., & Thagard, P. R. (1986) *Induction: processes of inference, learning and discovery.* MIT Press.

Kaelbling, L. P., Littman, M. L., & Moore, A. W. (1995) Reinforcement learning: a survey. In *Practice and Future of Autonomous Agents,* Vol. 1. Centro Stefano Franscini, Monte Verita, Ticino, Switzerland.

Laird, J. E., Newell, A., & Rosenbloom, P. S. (1987) SOAR: an architecture for general intelligence. *Artificial Intelligence* 33, 1-64.

McCarthy, J. (1979) Ascribing mental qualities to machines, *Philosophical Perspectives in Artificial Intelligence,* Ringle, M. (ed.), Harvester Press.

Newell, A. (1990) *Unified theories of cognition.* Harvard University Press, Cambridge, MA.

Oatley, K. (1992) *Best laid schemes, the psychology of emotions.* Studies in Emotion and Social Interaction, Cambridge University Press.

Oatley, K. & Johnson-Laird, P. N. (1985) Sketch for a cognitive theory of emotions, Cognitive Science Research Papers CSRP 045, University of Sussex.

Ortony, A., Clore, G., & Collins, A. (1988) *The cognitive structure of emotions.* Cambridge University Press.

Pfeifer, R. (1994) The 'Fungus Eater Approach' to emotion: a view from artificial intelligence. Extended and revised version of an invited talk at *AISB-91*, Leeds, U.K. Japanese version in: *Cognitive Studies*: Bulletin of the Japanese Cognitive Science Society, 1(2), 1994, 42-57. Also available as a technical report from AI Lab, Institute for Informatics, University of Zurich-Irchel.

Powers, W. T. (1988) *Living Control Systems* – selected papers of William T. Powers. The Control Systems Group, Kentucky.

Read, T., & Sloman, A. (1993) The terminological pitfalls of studying emotion. Paper presented at the *Workshop on Architectures Underlying Motivation and Emotion - WAUME 93*, Birmingham.

Riolo, R. L. (1988) CFS-C: A package of domain independent subroutines for implementing classifier systems in arbitrary, user-defined environments. Logic of Computers Group, Division of Computer Science and Engineering, University of Michigan.

Ryle, G. (1949) *The concept of mind.* Hutchinson.

Simon, H. A. (1967) 'Motivational and Emotional Controls of Cognition', reprinted in *Models of Thought*, Yale University Press, (1979) 29–38.

Simon, H. A. (1982) Comments, In M. S. Clark & S. T. Fiske (Eds.), *Affect and cognition.* Hillsdale, NJ: Lawrence Erlbaum Associates, 1982.

Simon, H. A. (1995) Artificial Intelligence: an empirical science. *Artificial Intelligence* 77 (1995) 95-127.

Sloman, A. (1987) 'Motives mechanisms and emotions' in Emotion and Cognition 1, 3, 217–234, reprinted in M. A. Boden (ed) *The Philosophy of Artificial Intelligence* "Oxford Readings in Philosophy" Series, Oxford University Press, 231–247 1990.

Sloman, A. (1992) Prolegomena to a theory of communication and affect, *Communication from an Artificial Intelligence Perspective: Theoretical and Applied Issues*, A. Ortony, J. Slack, and O. Stock, eds. Springer: Heidelberg, 229 – 260.

Sloman, A. (1993a) The mind as a control system. In *Proceedings of the 1992 Royal Institute of Philosophy Conference 'Philosophy and the Cognitive Sciences'*, (eds) C. Hookway and D. Peterson, Cambridge University Press.

Sloman, A. (1993b) Prospects for AI as the general science of intelligence. In *Prospects for Artificial Intelligence – Proceedings of AISB93*, Oxford IOS Press, Editors: A. Sloman, D. Hogg, G. Humphreys & D. Partridge.

Sloman, A. (1995) Exploring design-space and niche-space. In *Proceedings 5th Scandinavian Conf. on AI*, Trondheim May 1995, IOS Press, Amsterdam.

Sloman, A., Beaudoin, L.P., & Wright, I. P. (1994) Computational modeling of motive-management processes, *Proceedings of the Conference of the International Society for Research in Emotions*, Cambridge, July 1994. (ed) N.Frijda, ISRE Publications. 344-348.

Sloman, A., & Croucher, M. (1981) Why robots will have emotions. In *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, (197-202) Vancouver.

Steels, L. (1994) The artificial life roots of artificial intelligence. *Artificial Life Journal*, Vol 1, 1. MIT Press, Cambridge.

Strongman, K. T. (1987) *The Psychology of Emotion.* Third Edition, John Wiley & Sons.

Wierzbicka, A. (1992) Defining emotion concepts. *Cognitive Science* 16, 539-581.

Wilson, S. W. (1995) Classifier fitness based on accuracy. *Evolutionary Computation* 3(2) 149-185.

Wilson, S. W. & Goldberg, D. E. (1989) A critical review of classifier systems. In *Proceedings of the Third International Conference on Genetic Algorithms* (pp. 244-255), Los Altos, California: Morgan Kaufmann (1989).

Wright, I. P. (1995) *Cognition and Currency Flow*, notes toward a circulation of value theory of emotions. Unpublished research document, Cognitive Science Research Centre, University of Birmingham. (Available on request from the author.)

Wright, I. P., Sloman, A., & Beaudoin, L. P. (to appear) Towards a design-based analysis of emotional episodes. To appear, with commentaries, in *Philosophy Psychiatry and Psychology.*

LEARNING

Skinnerbots

David S. Touretzky
Computer Science Department &
Center for the Neural Basis of Cognition
Carnegie Mellon University
Pittsburgh, PA 15213-3891
dst@cs.cmu.edu

Lisa M. Saksida
Robotics Institute &
Center for the Neural Basis of Cognition
Carnegie Mellon University
Pittsburgh, PA 15213-3891
saksida@ri.cmu.edu

Abstract

Instrumental (or operant) conditioning, a form of animal learning, is similar to reinforcement learning in that it allows an agent to adapt its actions to gain maximally from the environment while only being rewarded for correct performance. But animals learn much more complicated behaviors through instrumental conditioning than robots presently acquire through reinforcement learning. We describe a new computational model of the conditioning process; our discussion focuses on a training technique called chaining. Four aspects of our model distinguish it from simple reinforcement learning: conditional reinforcers, shifting reinforcement contingencies, explicit action sequencing, and state space refinement. We apply our model to a task commonly used to study working memory in rats and monkeys: the DMTS (Delayed Match to Sample) task. Animals learn this task in stages. Our model also acquires the task in stages, in a similar manner. We have also used our learning program to control a B21 robot.

1 Introduction

A service dog trained to assist a handicapped person with the tasks of daily living can respond to over 60 verbal commands to turn on lights, open refrigerator doors, retrieve dropped objects, etc. [9]. Many other animals, such as rodents, pigeons, and dolphins, can also acquire complicated behavioral routines. (See [6, 22] for striking accounts of behaviors taught to a variety of species.) Animals can learn these tasks quickly and with robust results through the use of training techniques derived from knowledge of instrumental (or operant) learning.

Training techniques for mobile robots, such as reinforcement learning, have not demonstrated anywhere near the robustness and complexity of results of extant animal training techniques. While this disparity may be due in part to the superior perceptual and motor capabilities of animals, animal training has also been studied for considerably longer, and by many more investigators. We suggest that closer attention paid to the animal training literature, and a serious attempt to model the effects described there, may yield benefits of immediate value to robot learning researchers, and also provide a new, computationally-oriented perspective on animal learning.

Due to the pioneering work of B. F. Skinner in this area [8], we have coined the term "Skinnerbot" to describe an autonomous learning agent that employs strategies and exhibits behavioral effects characteristic of instrumental learning. The present paper describes the investigation of a particular conditioning technique called chaining — in which behavioral routines are built up from smaller action segments — and how it can be applied to mobile robot learning. We developed a learning algorithm that incorporates aspects of chaining which reinforcement learning techniques do not address, such as shifting reinforcement contingencies and learning of conditioned reinforcers. We chose a classic cognitive assessment task that involves behavioral sequences, the Delayed Match to Sample (DMTS) task [5], as the first test case for our learning model. We have also implemented the model on a B21 mobile robot.

1.1 Operant Conditioning

In operant conditioning, the acquisition and further performance of an action depends on the consequences experienced upon its completion. This type of learning is called "operant" because the behavior operates on (has an effect on) the environment; it is "instrumental" because the behavior is instrumental in producing reward. It is this type of learning that affords the animal some degree of control over its environment in that it has the ability to produce changes in its situation by performing an appropriate action. For example, the animal may learn that it can produce a desired stimulus, such as food, by pressing a lever. It follows that instrumental conditioning is one mechanism that enables an animal to cope with a dynamic environment in which the consequences of behaviors may vary.

This contrasts with classical, or Pavlovian, conditioning, in which learning is limited to associating a possibly arbitrary conditioned stimulus with a reinforcing (unconditioned) stimulus that elicits some type of innate behavioral response. For example, food as the unconditioned stimulus produces appetitive responses such as salivation; electric shocks pro-

duce fear and avoidance responses; and noxious stimuli, such as a puff of air delivered to the eyeball, produce defensive responses. In a classical conditioning procedure, an initially neutral stimulus such as a tone or light is repeatedly followed by an unconditioned stimulus. After learning, the conditioned stimulus comes to elicit a similar behavioral response, even in the absence of the unconditioned stimulus. Thus, when the bell rings the dog salivates, even if no food is delivered. Because the responses which occur during classical conditioning are innate, an animal that could only learn Pavlovian contingencies would be wholly dependent on evolutionary processes to construct appropriate responses to stimuli. In addition, in this paradigm the animal learns about stimuli, reinforcers, and the relationship between them, but learns nothing about the consequences of its own actions. As a result, if the consequences of a *response* somehow changed, the animal would be unable to adapt.

Pavlovian conditioning is of some value to robots: it is useful for a robot to be able to learn predictive values of stimuli and possibly follow them with innate anticipatory responses. This form of conditioning has a well-developed computational theory, the Rescorla-Wagner theory and its descendants [24, 33, 17, 2], that predicts the strength of a stimulus-reward association based on factors such as stimulus saliency, background stimulus rate, and training history. In addition, some simple models of classical conditioning have been implemented on robots [34]. But instrumental learning, in which an association between actions and their *outcomes* is built, allows for the modification of responses in an unstable environment; it confers an ability that is probably more critical to the robustness and practicality of a mobile robot. At present, there are no theories of instrumental conditioning comparable in scope and explicitness to the Rescorla-Wagner model of classical conditioning. The goal of our work is to provide such a theory, instantiated as a computational model. The present paper describes an initial step in that direction.

1.2 Chaining

Complex behaviors can often be broken down into components and analyzed as a sequence of operants.[1] For example, a chick trained to "play the piano" pecks a sequence of keys to obtain a food reinforcement at the end of the tune [6]. A pig taught to "grocery shop" pushes a cart and selects specific items to place in it, one after the other [6].

A behavioral chain can be analyzed as a sequence of stimuli and responses. The core unit of a chain is called a *link*; it consists of a discriminative stimulus, a response, and a reinforcer. The chain begins with the presentation of the first discriminative stimulus. When the animal makes the appropriate response in the presence of this stimulus, a conditioned reinforcer[2] is presented as a reward for the response. The reinforcer also functions as a discriminative stimulus for the next link in the chain, setting the occasion for the next desired response. This process continues for a number of links until reaching the final stimulus in the chain, which is a primary (innate) reinforcer such as food. The links are "overlapped" in that the discriminative stimulus for the production of one response is the reinforcer for the previous response; this holds the chain together. The concept of chaining differs from other examples of response sequences, such as fixed action patterns, in that chains of behavior can be modified through reinforcement. Fixed action patterns, as found in many animals, are hardwired: once the sequence is initiated it goes to completion independent of the consequences of the behavior. An example of this type of behavior occurs in the Greylag Goose. When an egg rolls out of its nest it will stand up, put its bill on the egg, pull back toward its chin, and roll the egg into its nest. While engaged in this fixed action pattern, the goose always performs the same behaviors in the same order, it will continue the pattern even if it loses its grip on the egg, and the pattern is triggered by any round stimulus outside its nest (including beach balls). (See [1] for more examples of fixed action patterns.) Thus this type of behavior is much less flexible than that involved in instrumental chaining.

The idea that patterns of responding can be reduced to a succession of stimulus-response units has been controversial: Skinner [30] claimed that all behavior, including language, could be represented this way, while others, such as Chomsky [10] and Lashley [19] held that sequential behavior could not be adequately accounted for in these terms. There is now considerable evidence, however, that many, though probably not all types of behavior sequences are held together this way [12].

The concept of constructing behavioral sequences for mobile robots from small elements is appealing in that the programmer's responsibility would be limited to the construction of just these behavioral primitives, plus the learning algorithm for putting them together. Many different behaviors could be assembled from a well-designed set of primitives, and learning could potentially be made faster because knowledge could be shared among tasks with similar sub-tasks.

1.3 Previous models

Only a few previous computational models of operant conditioning phenomena have been described. Models of conditioning in *Aplysia* [3, 23] have focused on learned suppression of a motor action. Mixed classical/operant models (known as "two process models") of escape and avoidance behavior in vertebrates [14, 26] address only simple responses to con-

[1] The animal learning literature defines at least two classes of behavioral responses [27]: (i) respondents, which originate with the stimuli that elicit them (e.g., a reflex), and (ii) operants, which are determined by their effects on the environment since they do not require eliciting stimuli.

[2] At least two types of reinforcers can be distinguished [25]: (i) primary reinforcers can reinforce behavior without the animal having had any prior experience with them (e.g., food, water). (ii) conditioned reinforcers acquire the power to reinforce behavior during the lifetime of the animal via a Pavlovian mechanism in which the stimulus that becomes the conditioned reinforcer is repeatedly paired with a primary reinforcer.

ditioned stimuli. None of these models approach the full richness of vertebrate learning, involving, for example, acquisition of secondary reinforcers and construction of behavior chains.

Graham, Alloway and Krames [13] describe a "virtual rat" designed to let undergraduates try their hand at operant conditioning. Their program is hard-wired to acquire a particular conditioned reinforcer (the sound of a food dispenser) and to respond to a specific shaping strategy to teach the simulated rat to bar press for food [18]. Other primitive actions, such as grooming, can be encouraged by linking them to food rewards, but the program isn't flexible enough to permit shaping anything complex except for bar pressing, nor is it possible to teach the rat to respond to external signals such as a tone or light. There are also presently no provisions for chaining behaviors, for modifying the qualities of a particular motor response, or for refining the animal's perceptual abilities.

Maki and Abunawass [21] model learning of a match-to-sample task (no delay) using a backpropagation network. In animals, this task requires learning a complex sequence of actions (see section 1.4). Maki *et al.*'s network, however, takes a sample stimulus and two potential match stimuli as input, and learns to compute two exclusive-OR functions for its output. It produces no overt behavior, just a "match left" or "match right" signal. Thus, the model does not emulate operant conditioning, but it does offer some suggestions about the learned internal representations of stimuli that might result from such conditioning.

Reinforcement learning also bears some similarity to operant conditioning, since reinforcement learning techniques do not need to be shown correct responses as training stimuli, as is required by supervised learners such as backpropagation. Like operant conditioning, reinforcement learning is appealing because it theoretically allows an agent to adapt its actions to get the most from its environment as it gains information over time. In practice, however, most RL applications focus on single tasks. Some work has been done on sequential task learning, but it tends to be limited to small state and operator spaces. Singh [29] describes a sequential task learner in which separate "Q-modules" learn different elemental and composite tasks. Mahadevan and Connell [20] use Q-learning to acquire multiple behaviors that could then be controlled using a hard-wired switching scheme to designate which should be active at a given time. Although both of these papers look at sequential task learning, their approaches have been demonstrated only in very simple environments, since they are subject to the usual combinatorial limitations of Q-learning. Also, a fairly large amount of knowledge had to be built into both systems: Singh's approach requires a Q-module to be designed in advance for each elemental task, and Mahedevan *et al.'s* system incorporates a hardwired behavior switching mechanism.

1.4 The DMTS Task

The Delayed Match to Sample task is widely used in cognitive neuroscience to measure properties of working memory. The basic idea is to present the animal with a stimulus (the sample), impose a delay, and then present a pair of stimuli, one of which matches the sample. The animal must select the matching stimulus in order to receive a reward. The delay period can be varied to control the length of time the sample must be maintained in working memory. There are both spatial and nonspatial versions of this task. In the spatial version, all stimuli are identical; they are distinguished on the basis of the location at which they appear [16]. In the nonspatial version, the sample appears in one location and the two probe stimuli appear in other locations; it is the visual characteristics of the stimuli that matter [7]. There is some debate as to whether the spatial version of the task actually involves working memory; animals might be using some other type of mediating strategy, such as aligning their body toward the site of the last stimulus, to bridge the delay [15]. Our learning algorithm can be applied to either the spatial or non-spatial version of the task.

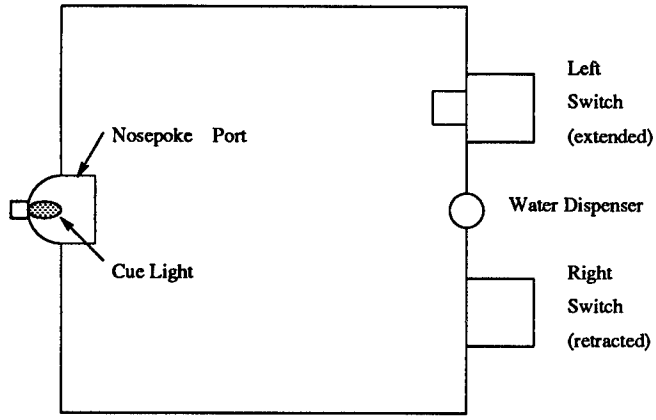


Figure 1: Skinner box configured for DMTS task.

Hampson, Heyser, and Deadwyler [16] describe a spatial version of DMTS for rats that uses as stimuli two retractable switches mounted on the wall of a Skinner box, as shown in Figure 1. A water dispenser is located between the two switches, and a light and nosepoke port are mounted on the opposite wall. At the start of a trial, one of the switches extends, and the rat must go over and press that switch. This causes the switch to retract, while at the same time a light goes on over the nosepoke port. The rat must now go to the opposite wall and make repeated nosepokes for a variable delay period averaging one minute. (The nosepoke requirement is intended to prevent the rat from parking itself in front of the switch it just pressed until the switches extend again. That sort of mediating strategy would eliminate the need for working memory.) At the conclusion of the delay period, the next nosepoke causes the light to go out and both switches to extend. Now the rat must return to the switch it pressed previously and press it again. If it chooses the correct switch, it receives a water reward.

Rats are taught this task in stages. Hampson et al. report a training time of two to three months.

2 Implementation

2.1 A Model of Conditioning

A close examination of the steps involved in the chaining of animal behavior reveals several important issues that are critical to the success of the procedure, and which have not been considered in previous computational models of conditioning:

Shifting Reinforcement Contingencies: In operant conditioning there is a nonstationary reward function: the trainer changes the criteria for success over time to gradually shift the animal's behavior closer to the desired behavior, in a process known as "shaping." There are also gross changes in reward contingencies each time a new phase of training is begun in the construction of a complex behavioral sequence. Most reinforcement learning algorithms, for example Q-learning [35], can track nonstationary environments, but they do not detect nonstationarity and explicitly respond to it, as animals do. Our learning mechanism does not require information about the structure of the learning task to be built in: it detects any sudden change in reinforcement contingencies and then begins to explore its action space, looking for potential new predictors of reward. Meanwhile, it maintains previously learned knowledge.

A more dramatic example of response to a change in reward is the phenomenon known as "extinction." When reinforcement of a behavior is discontinued, the animal will eventually stop producing that behavior. But in the short term its activity level actually rises in response to discontinued reinforcement, and the variability of its responses also increases. In this way, it broadens its exploration of the action space and may discover a variant of the learned action that will once again produce the expected reward. We are presently working on adding extinction phenomena to our model.

Conditioned Reinforcers (Bridging Stimuli): Contiguity of action and outcome are critical to instrumental learning: an action must be closely followed by a reinforcer in order for the animal to learn an association between the two. In a training situation, however, it is often difficult to reward an animal with food immediately after the occurrence of the desired response. Conditioned reinforcers are stimuli that become associated with food, and serve as a signal that "food is coming," thereby eliminating the gap between the desired action and the reinforcement signal. For example, in a Skinner box, every time the animal is about to receive a food pellet it will hear the click of the food dispenser operating. The sound soon becomes a conditioned reinforcer; the animal learns that the click means food will soon be available, and the close temporal contiguity of an action and the sound of imminent reward is sufficient to produce a much increased likelihood of performance of that action.

Since the sound of the dispenser can be heard throughout the Skinner box, the animal can now be rewarded when it is not at the food hopper.

In typical RL techniques, credit assignment is a major problem: after completing a sequence of actions which led the agent to the goal, in order to learn which of those actions should be credited with contributing to the final success, the agent needs to evaluate the goodness of each action that was performed. When the only reward obtained occurs at the end of the action sequence, only knowledge of the cumulative effect of the actions can be derived. Our model deals with this issue by learning conditioned reinforcers of the sort described above. It discovers primitive subgoals, such as hearing the sound of the dispenser activating, and seeks ways to achieve them.

Action Sequencing: One difficulty with sequential task decomposition is that a mechanism must be in place for directing the construction of the behavior chain: it is highly unlikely that an agent will be able to achieve a complicated goal state simply by composing action sequences randomly. Our model uses an explicit temporal predicate representation that allows us to distinguish the order in which events occur in order to learn behavioral sequences. The primitive subgoals, mentioned above, also function as discriminative stimuli and thus set the occasion for an action to take place.

Perceptual (State Space) Refinement: A further problem with RL techniques is that they are usually restricted to a very small state space to avoid combinatorial explosion. As in other approaches based on explicit symbolic representations (including much of classical AI), we accomodate a large state space by factoring it into sets of predicates. We can then construct logical expressions to refer to collections of states in economical ways. Another advantage of this approach is that it permits incremental refinement of the state space by adding new predicates. For example, an animal will learn to distinguish tones at two different frequencies if they are associated with different rates of reward. Thus, a predicate like `HEAR(tone)` might eventually be supplanted by `HEAR(high-tone)` and `HEAR(low-tone)`. We have not included perceptual shaping in our current learning algorithm, but intend to address it in the future.

2.2 Memory Representation

To introduce our model, we consider the case of a rat pressing a switch to get water. The working memory module (WM) holds a collection of time-labeled predicates describing the rat's current perceptions and actions and those of the recent past. For example, the label ":2" below means the item occurred two timesteps ago. At the instant when the rat receives a water reward, having previously heard the pump run, the contents of WM might look like this:

AT(ne-corner):2 GOTO(se-corner):2
AT(se-corner):1 HEAR(pump):1 GOTO(dispenser):1
AT(dispenser):0 RECEIVE(water) :0

The algorithm for inferring reinforcement contingencies operates on a collection of slightly more abstract items called *temporal predicates*. These are derived from WM elements by replacing the label `:0` by `:now`, the label `:1` by `:prev` and all positive labels (including 1) by `:past`. For conciseness, `:now` tags will usually be left implicit in the rest of this paper. There is also a `:fut` tag for referring to predicates that became true at $t+1$ during retrospective analysis of the results of action at time t. Working memory elements persist for only a small number of time steps, depending on the predicate involved. In the present simulation, `AT` and `GOTO` predicates last for two time steps, whereas `SEE`, `HEAR`, and `PRESS` last for six. This is because the animal is always at some location and nearly always moving, whereas conditioned stimuli and specalized actions (such as switch pressing) occur less frequently, and so are more memorable.

In the next level of representation in the program, we form conjunctions of predicates. Temporal tagging of these items allows us to infer cause and effect relationships between actions and stimuli, and construct temporal sequences. For example, the crucial match relationship in the DMTS task is described by the conjunction `PRESS(sw1):past & PRESS(sw1):now`, plus a similar conjunction for the second switch.

2.3 *Learning Reinforcement Contingencies*

The conjunctions our program constructs describe sequences of stimuli and actions that can occur in the world. Some of these occur frequently; others might never be encountered. Furthermore, some sequences are often followed by a reinforcement signal whereas others never are. In order to extract this information from its experience of the world, the program maintains two tables for each reinforcer. One counts the number of times each conjunction has occurred in WM since that reinforcer was acquired; the other table counts the number of times a conjunction's occurrence has been followed on the next time step by the reinforcer. The *reward rate* of a conjunction is the second quantity divided by the first. The program attempts to find conjunctions with maximum reward rates. A conjunction that predicts rewards with no false alarms would have a reward rate of 1.0.

In complex domains we cannot afford to test all possible conjunctions of predicates, so heuristic search is used. Conjunctions are constructed incrementally by combining a pool of currently "best" conjunctions (starting with the null conjunction) with a pool of "best" predicates. A "best" conjunction is one whose reward rate is at least one standard deviation above the mean rate, or whose reward count is at least one standard deviation above the mean count. Both tests are necessary. Items with high reward *counts* constitute important features of the environment that need to be incorporated into conjunctions even if their reward rates in isolation are low. For example, since going to the water dispenser does not make the pump run, `GOTO(dispenser)` has a low reward rate, but since water is available nowhere else, it has a high reward count. Items with high reward *rates* are accurate predictors and should be retained for further exploration, even if their reward *counts* are relatively low (meaning they each account for only a limited number of the rewards that have been received.)

During learning, conjunctions that are sufficiently well correlated with rewards generate "predictors," i.e., rules for predicting reward. These may displace earlier predictors that haven't performed as well. During initial *magazine training* (learning to go to a food or water dispenser when the mechanism is heard to activate), a typical sequence of learned predictors and their reward rates is shown below.

1: RECEIVE(water) ← GOTO(dispenser) [0.154]
2: RECEIVE(water) ← HEAR(pump) [0.3]
3: RECEIVE(water) ← HEAR(pump) & GOTO(dispenser) [1.0]

Predictor # 3 says that 100% of the time, when the simulated rat heard the pump and immediately went to the water dispenser, it received a water reward. So this conjunction predicts water with perfect accuracy.

To generate behavior, we look for predictors that can be satisfied by the rat's taking some action currently available to it. Predictor # 1 suggests going to the water dispenser, so initially the rat spends a lot of time there. This causes the reward rate of `GOTO(dispenser)` to drop, since on most occasions there will be no water there. (But the reward count for the predictor remains high relative to all other predicates, since the dispenser is still the only place where water can be obtained.) Because predictor # 1 gives many false expectations of reward, it is soon dropped. Predictor # 2 is somewhat more successful, but it cannot be satisfied by the rat's own actions, because at this point in training the rat has no way to cause the pump sound to occur. But predictor # 3, which most accurately describes the current reward contingencies in this environment, *can* be used by the rat to generate behavior whenever the pump is heard to run.

2.4 *Acquiring Conditioned Reinforcers*

The second type of learning in our model is the acquisition of new conditioned reinforcers. If the Skinnerbot could find a way to make `HEAR(pump)` be true, then predictor # 3 suggests it could get water whenever it wanted. So `HEAR(pump)` becomes a secondary reinforcer, and the Skinnerbot begins trying out theories of what causes the pump to run. At about this point in the training process, the human trainer stops randomly triggering the pump for magazine training and only rewards switch presses. By exploration, the Skinnerbot eventually discovers that pressing the switch will make the pump run, and nothing else will. Thus, the following predictor is acquired:

#7: HEAR(pump) ← PRESS(switch) [1.0]

Now suppose the Skinnerbot is at the water dispenser along the north wall of the Skinner box, but the switch is on the south wall. Predictor #7 cannot apply because the `PRESS(switch)` operator is not available at the north wall. This impasse generates a new secondary reinforcer, `CAN(PRESS(switch))`, which the Skinnerbot also seeks to control, leading it to discover another predictor:

#8: CAN(PRESSS(switch)) ← AT(south-wall):fut [1.0]

The `:fut` tag indicates that if the Skinnerbot is at the south wall at time $t + 1$ it will be able to press the switch at $t + 1$. It can make the predicate true by a `GOTO` action, or if it's already at the south wall, it need only refrain from going elsewhere.

Now the Skinnerbot has a hierarchy of reinforcers. The primary (innate) reinforcer is water. The most important secondary reinforcer is the pump sound. A more remote secondary reinforcer is the ability to press the switch.

2.5 *Backward Chaining*

At each time step, the Skinnerbot seeks a predictor it can satisfy. Predictors are prioritized by the nature of the reinforcement they promise, so that given a choice, the Skinnerbot will always act to secure a more basic reward (water) over a more abstract one (the ability to press the switch.) If it finds a predictor where all but one of the predicates is currently true (i.e., matches an item in WM), and the last one can be made true by taking some action that is presently available, then it will select that action with high probability. (There is some randomness in the system to facilitate continued exploration.)

With predictors #3, 7, and 8, plus the ordering imposed by the reinforcer priorities, the Skinnerbot will repeatedly shuttle between the north and south walls, pressing the switch when at the south wall, hearing the pump, and collecting its water reward at the north wall.

`CAN` goals must be handled specially. They are only looked at when the program has some other subgoal that could be satisfied if the action were available. For example, with a predictor whose antecedent is `SEE(light)` & `POKE(poke-port)`, the Skinnerbot only needs to be able to poke if it sees the light; the rest of the time it doesn't matter whether it can poke or not. When `SEE(light)` is true, the predictor could be satisfied if `POKE(poke-port)` were an available action. Hence `CAN(POKE(poke-port))` becomes a goal worth satisfying, and this in turn allows predictor #8 to lure the Skinnerbot to the location of the poke port.

2.6 *Shifting Reinforcement Contingencies*

Rats go through several training stages in learning the DMTS task, first pressing a switch to get a water reward, then learning to nosepoke when the light is on to get a switch to extend. The next stage requires the rat to learn to press a switch in order to turn on the light above the nosepoke port. At this point the meaning of a switch press has changed. Before it produced only a water reward; now it can produce either water or a light, depending on context. Predictors that earlier did an adequate job of characterizing the environment's reinforcement contingencies must now be replaced with more selective versions. Our model responds to the changed situation by adjusting the reward tables that govern its behavior, which has the effect of making the model more plastic – eager to acquire new predictors and more willing to replace old ones.

Because the model is always trying to formulate explanations for reinforcers, it will construct them even when no explanation is possible. For example, during magazine training the pump is triggered at random times by the experimenter. The model will generate predictors for `HEAR(pump)` and act on whichever ones have the highest apparent reward rate. Thus, if on several occasions it happened to be moving from the southeast corner to the northeast corner when the pump ran, it might decide that this action was *causing* the pump to run, and begin repeating it deliberately. If water is actually being dispensed randomly, but at sufficiently frequent intervals, the predictor will be successful often enough to be retained and perhaps even strengthened. This sort of "superstitious" behavior has been observed in real animals [31] although the underlying mechanisms are at this point unclear [32].

At the next stage of training, where a switch is presented and switch pressing reliably causes the pump to run, the earlier superstitious predictors are quickly supplanted by the more effective predictor #7.

3 Results

3.1 *DMTS Simulation*

A minimum of ten predictors are required for the DMTS task, as shown in Figure 2. Their acquisition is sensitive to factors such as the model's working memory capacity for predicates and the thresholds for predictor creation and deletion. To reach this rule set the learner had to go through a number of intermediate stages to acquire component behaviors and set up the necessary secondary reinforcers. The program generated many other predictors in the course of the simulation. Most of these were eventually replaced by better-preforming predictors; some were retained. For example, one predictor for `HEAR(pump)` contains a redundant `SEE(light):past` clause. Some of the additional predictors acquired by the program are shown in Figure 3. These are not always correct on their own, but they generally support rather than hinder the correct predictors.

The `NOT(reinforced):past` predicate in predictors 206 and 215 is used to distinguish the first appearance of a switch (where pressing it turns the light on) from the second appearance (where a press should result in the pump sound.) At the start of a new trial, assuming a minimally reasonable inter-trial interval, there are no reinforcers in working memory. The predicate above expresses that fact; it serves as

a contextual marker for the start of a trial. For the second switch press of the trial, the earlier switch press provides the context, as in predictors 205 and 210.

A sample run of our learning program on the DMTS task is shown in Figure 4. At time step 5376 the simulated rat is waiting for a new trial to begin. At 5377 a switch appears, and at 5378 the rat presses it. At time step 5379 predictor 54 erroneously takes the rat to the dispenser when it should have gone to the poke port, but on the next time step predictor 107 takes it to the poke port via subgoal predictor 130, and it nose pokes at time 5381. The match response takes place at 5382-5383, and the trial concludes at 5385 with the receipt of a water reward.

3.2 Robot Implementation

Amelia is an RWI B21 mobile robot with a color camera on a pan-tilt head and a three degree-of-freedom arm with gripper. Computing power is provided by two on-board Pentium processors plus a laptop Pentium with color display. A 1 Mbps radio modem links Amelia to a network of high-end workstations that can contribute additional processing cycles. For our experiments, we ran the learning program in Allegro Common Lisp on a Sparc 5 and used Reid Simmons' Task Control Architecture [28] to communicate with the robot.

To provide reinforcement and bridging stimuli to the robot, we added a Logitech three-button radio trackball. The human trainer can stand anywhere in the vicinity of the robot and press a button to send a reward signal when a desired response occurs. The button press is picked up by an on-board receiver plugged into a serial port on one of the Pentium boards and relayed to the learning program on the Sparc. A second button is used to simulate a neutral sensory input, such as a light or tone. The robot acknowledges button presses with a brief audio response.

Our first experiment teaching behaviors to Amelia avoided the use of sensory input. We decided to begin by reinforcing spontaneous gestures. The robot was given an innate "body language" consisting of several types of arm movements programmed to occur at low but nonzero free operant rates. We taught Amelia to selectively make a "wave," "clap," or "salute" gesture by rewarding the preferred gesture when it occured. In a second experiment, we produced a two-gesture sequence by first rewarding the robot for waving, then teaching it that waves would only be rewarded when they occur in response to an external stimulus: a button press. Finally we taught it that clapping would result in a button press being received. The button press serves as the bridging stimulus: it is a conditioned reinforcer. When training real animals a hand-operated clicker is sometimes used to provide this stimulus.

These were only initial experiments, but in order to get even this simple system to work on the robot we had to solve a variety of hardware and software interface problems. With this infrasturcture in place, we plan to add a crude visual perception facility in order to demonstrate the full capabilities of the learning model on the robot.

4 Discussion

We have described a model of an operant conditioning technique called chaining in which behaviors are progressively combined in order to yield more complicated action sequences. Our model focuses on four aspects of chaining that differentiate it from other computational models of operant conditioning.

1. Reinforcement contingencies change over time. The incorporation of this into our model allows a trainer to continuously add new elements to the animal's (or robot's) behavior without losing previously learned information. This mechanism thus allows an agent to adapt to a dynamic environment in which the actions leading to reward may not remain constant.

2. Conditioned reinforcers hold the chain together. These learned reinforcers consist of perceptions that naturally become associated with the completion of an action as long as they occur consistently at that time. Conditioned reinforcers help deal with the problem of credit assignment without the experimenter having to go through too many trials to teach a complex task.

3. Discriminative stimuli "set the occasion" for performing a particular behavior. In other words, they signal to the agent that execution of an action will now produce a reward, such as when a dog rolls over in response to a hand signal. In chained behaviors, discriminative stimuli help the animal keep track of the order of actions. An example is the light in the DMTS task that indicates it is time to nose poke.

   Discriminative stimuli may change over time. Animal trainers use a technique called "stimulus fading," where the signal to perform an action is gradually made more subtle, or perhaps replaced altogether with a less salient stimulus.

4. Finally, a major problem with reinforcement learning techniques is that they usually represent states as discrete table entries, so combinatorial considerations tend to limit them to small state spaces. In this paper, we factor state space into collections of propositions that refer to collections of states in an economical way, as do many other AI programs. What is novel here is the development of heuristics based on reward rate and total reward to guide the construction of these expressions based on the agent's training experience.

Much work remains to be done to complete a computational-level model of operant conditioning. In order to expand our model of chaining to incorporate more results from the animal learning literature, one idea that we would like to explore is operator shaping. We would like to equip our robot with an initial set of innate behaviors (operators) which

#24: RECEIVE(water) ← HEAR(pump) & GOTO(dispenser)

#210: HEAR(pump) ← SEE(light):past & PRESS(switch1):past & PRESS(switch1)
#205: HEAR(pump) ← PRESS(switch2):past & PRESS(switch2)

#206: SEE(light) ← NOT(reinforced):past & PRESS(switch1)
#215: SEE(light) ← NOT(reinforced):past & PRESS(switch2)

#68: CAN(PRESS(switch1)) ← SEE(switch1) & GOTO(sw1-loc)

#81: CAN(PRESS(switch2)) ← SEE(switch2) & GOTO(sw2-loc)
#115: SEE(switch1) ← SEE(light) & POKE(poke-port)

#107: SEE(switch2) ← SEE(light) & POKE(poke-port)

#95: CAN(POKE(poke-port)) ← AT(port-loc):fut

Figure 2: Essential predictors learned in the DMTS task.

#30: RECEIVE(water) ← HEAR(pump) & AT(dispenser):fut

#197: SEE(light) ← GOTO(poke-port):past & PRESS(switch1)

#226: SEE(switch1) ← SEE(light):prev & GOTO(poke-port):prev & AT(poke-port):fut

#138: SEE(switch2) ← NOT(reinforced):past & GOTO(poke-port):prev & SEE(light)

#54: CAN(PRESS(switch2)) ← SEE(switch2):prev & AT(dispenser)

Figure 3: Some of the additional predictors learned in the DMTS task. Predictor 54 is incorrect, but does not cause serious problems for the program.

5376: NOT(reinforced) AT(sw1-loc) -C-> GOTO(dispenser)
5377: SEE(switch2) AT(dispenser) -81-> GOTO(sw2-loc)
5378: CAN(PRESS(switch2)) SEE(switch2) AT(sw2-loc) -215-> PRESS(switch2)
5379: SEE(light) AT(sw2-loc) -54-> GOTO(dispenser)
5380: SEE(light) AT(dispenser) -130-> GOTO(port-loc)
5381: CAN(POKE(poke-port)) SEE(light) AT(port-loc) -107-> POKE(poke-port)
5382: CAN(POKE(poke-port)) SEE(switch2) SEE(switch1) AT(port-loc) -81-> GOTO(sw2-loc)
5383: CAN(PRESS(switch2)) SEE(switch2) SEE(switch1) AT(sw2-loc) -205-> PRESS(switch2)
5384: HEAR(pump) AT(sw2-loc) -24-> GOTO(dispenser)
5385: AT(dispenser) RECEIVE(water) -C-> GOTO(dispenser)

Figure 4: Sample run: one trial of the DMTS task. The number embedded in each arrow is the predictor being applied; a "C" indicates no predictor is satisfied in the present situation (clueless).

may not necessarily be able to fully satisfy the requirements of the trainer or the environment. What would then be needed is a means for refining operators with experience, similar to the animal training technique called "shaping". Evidence for the existence of innate behavioral elements [4] combined with the success of shaping in animal learning paradigms suggests that this would be a very powerful mechanism for improving the performance of our learning algorithm.

We also need to add facilities for refining perceptual predicates with experience, so that the model can acquire finer-grain discriminations if the reinforcement contingencies require this. Animals can learn to make very fine distinctions in pitch, intensity, and color, but they can also generalize on these properties, depending on the demands of the task. This makes state space dynamically refinable with experience.

We have coined the term "Skinnerbot" to refer to a class of agents designed for operant conditioning, but unlike Skinner, we do not eschew representations in our theory. Once we have laid more of the computational-level groundwork for our model, we will be able to move on to a model which addresses some of the presently unsettled psychological issues in instrumental learning [11].

References

[1] S.A. Barnett. *Modern Ethology*. Oxford University Press, 1981.

[2] A. G. Barto and R. S. Sutton. Time-derivative models of Pavlovian conditioning. In M. Gabriel and J. Moore, editors, *Learnng and Computational Neuroscience: Foundations of Adaptive Networks*, pages 497–537. MIT Press, Cambdridge, MA, 1990.

[3] D. A. Baxter, D. V. Buuonomano, J. L. Raymond, D. G. Cook, F. M. Kuenzi, T. J. Carew, and J. H. Byrne. Empirically derived adaptive elements and networks simulate associative learning. In *Neural Network Models of Conditioning and Action*, pages 13–52. Lawrence Erlbaum Associates, Hillsdale, NJ, 1991.

[4] K.C. Berridge, J.C.Fentress, and H. Parr. Natural syntax rules control action sequence of rats. *Behavioural Brain Research*, 23:59–68, 1987.

[5] D.S. Blough. Delayed matching in the pigeon. *Journal of the Experimental Analysis of Behavior*, 2:151–160, 1959.

[6] K. Breland and M. Breland. The misbehavior of organisms. *Ameican Psychologist*, 16:681–684, 1961.

[7] T. J. Bussey, J. L. Muir, and T. W. Robbins. A novel automated touchscreen procedure for assessing learning in the rat using computer graphic stimuli. *Neuroscience Research Communications*, 15(2):103–109, 1994.

[8] A. C. Catania and S. Harnad, editors. *The Selection of Behavior*. Cambridge University Press, 1988.

[9] CCI. *The CCI Program*. Canine Companions for Independence, Santa Rosa, CA, 1995. Informational page available at http://grunt.berkeley.edu/cci/cci.html.

[10] N. Chomsky. Review of Skinner's Verbal Behavior. *Language*, 35(26-58), 1959.

[11] A. Dickinson. Instrumental conditioning. In N. J. Mackintosh, editor, *Handbook of Perception and Cognition. Volume 9*. Academic Press, Orlando, FL, 1995.

[12] L. Gollub. Conditioned reinforcement: Schedule effects. In W.K. Honig and J.E.R. Staddon, editors, *Handbook of operant behavior*. Prentice-Hall, 1977.

[13] J. Graham, T. Alloway, and L. Krames. Sniffy, the virtual rat: Simulated operant conditioning. *Behavio Research Methods, Instruments, & Computers*, 26(2):134–141, 1994.

[14] S. Grossberg. A neural theory of punishment and avoidance, II: Quantitative theory. *Mathematical Biosciences*, 15:253–285, 1972.

[15] S.A. Gutnikov, J.C. Barnes, and J.N.P. Rawlins. Working memory tasks in five-choice operant chambers: use of relative and absolute spatial memories. *Behavioral Neuroscience*, 108(5):899–910, 1994.

[16] R. E. Hampson, C. J. Heyser, and S. A. Deadwyler. Hippocampal cell firing correlates of delayed-match-to-sample performance in the rat. *Behavioral Neuroscience*, 107(5):715–739, 1993.

[17] A. H Klopf. A neuronal model of clasical conditioning. *Psychobiology*, 16:85–125, 1988.

[18] L. Krames, J. Graham, and T. Alloway. *Sniffy, the Virtual Rat*. Brooks/Cole, Pacific Grove, CA, 1995. Includes software diskette.

[19] K. Lashley. The problem of serial order in behavior. In L.A. Jeffries, editor, *Cerebral mechanisms in behavior*. John Wiley and Sons, 1951.

[20] S. Mahadevan and J. Connell. Automatic programing of behavior-based robots using reinforcement learning. *Artificial Intelligence*, 55:311–365, 1992.

[21] W. S. Maki and A. M. Abunawass. A connectionist approach to conditional discriminations: Learning, short-term memory, and attention. In *Neural Network Models of Conditioning and Action*, pages 241–278. Lawrence Erlbaum Associates, Hillsdale, NJ, 1991.

[22] K. Pryor. *Lads Before the Wind*. Harper and Row, New York, 1975.

[23] J. L. Raymond, D. A. Baxter, D. V. Buonomano, and J. H. Byrne. A learning rule based on empirically derived activity-dependent neuromodulation supports operant conditning in a small network. *Neural Networks*, 5(5):789–803, 1992.

[24] R. A. Rescorla and A. R. Wagner. A theory of Pavlovian conditioning: Variations in the effectiveness of reinforcement and nonreinforcement. In A. H. Black and W. F. Prokasy, editors, *Classical Conditioning II: Theory and Research*. Appleton-Century-Crofts, New York, 1972.

[25] G.S. Reynolds. *A Primer of Operant Conditioning*. Scott, Foresman, and Company, 1968.

[26] N. A. Schmajuk and D. W. Urry. The frightening complexity of avoidance: An adaptive neural network. In *Models of Action*. Lawrence Erlbaum Associates, Hillsdale, NJ, 1995.

[27] B. Schwartz. *Psychology of Learning and Behavior*. W.W. Norton, 1989.

[28] R. Simmons. Structured control for autonomous robots. *IEEE Transactions on Robotics and Automation*, 10(1):34–43, 1994.

[29] S.P. Singh. Transfer of learning across sequential tasks. *Machine Learning*, 8:323–339, 1992.

[30] B.F. Skinner. *Behavior of Organisms*. Appleton-Century-Crofts, 1938.

[31] B.F. Skinner. "Superstition" in the pigeon. *Journal of Experimental Psychology*, 38:168–172, 1948.

[32] J.E.R. Staddon and V.L. Simmelhag. The "superstition" experiment: A reexamination of its implications for the principle of adaptive behavior. *Psychological Review*, 78:3–43, 1971.

[33] R. S. Sutton and A. G. Barto. Toward a modern theory of adaptive networks: Expectation and prediction. *Psychological Review*, 88:135–170, 1981.

[34] P. F. M. J. Verschure, J. Wray, O. Sporns, G. Tononi, and G. M. Edelman. Multilevel analysis of classical conditioning in a real world artifact. *Robotics and Autonomous Systems*, in press.

[35] C. J. C. H. Watkins. *Learning from Delayed Rewards*. PhD thesis, Cambridge University, Cambridge, England, 1989.

No Bad Dogs: Ethological Lessons for Learning in Hamsterdam*

Bruce M. Blumberg
MIT Media Lab
E15-305, 20 Ames St.
Cambridge Ma. 01463
bruce@media.mit.edu

Peter M. Todd
Max Planck Inst. for Psychological Research,
Center for Adaptive Behavior and Cognition
Leopoldstrasse 24, 80802 Munich Germany
ptodd@mpipf-muenchen.mpg.de

Pattie Maes
MIT Media Lab
E15-305, 20 Ames St.
Cambridge Ma. 01463
pattie@media.mit.edu

Abstract

We present an architecture for autonomous creatures that allows learning to be combined with action selection, based on ideas from ethology. We show how temporal-difference learning may be used within the context of an ethologically inspired animat architecture to build and modify portions of the behavior network, and to set fundamental parameters including the strength associated with individual Releasing Mechanisms, the time course associated with appetitive behaviors, and the learning rates to be used based on the observed reliability of specific contingencies. The learning algorithm has been implemented as part of the Hamsterdam toolkit for building autonomous animated creatures. When implemented in Silas, a virtual dog, the algorithm enables Silas to be trained using classical and instrumental conditioning.

1 Introduction

Action selection and learning represent two significant areas of research in behavior-based AI [Maes94], and advances in both areas are essential if we are to build animats that demonstrate robust adaptive behavior in dynamic and unpredictable environments. However, most work in this area to date has focused on one problem or the other. Several researchers [Maes90a, Blumberg94, Blumberg95, Tyrrell94, Tu94] have proposed ethologically inspired models of action selection that purport to handle many of the challenges associated with building animats which must juggle multiple goals at one time. However, this work did not incorporate learning, and the proposed architectures generally involved hand-built behavior networks and "not-a-few" parameters that had to be tweaked. In contrast, there has been a great deal of impressive work in learning [Watkins89, Whitehead92, Kaelbling92, Lin92, Mahadevan91, Sutton91], but the focus of this work has been on single-goal systems and on issues of optimality. Moreover, while it is now understood how to learn an optimal policy in certain kinds of worlds, it is also recognized that this is just the tip of the iceberg and that addressing the issues of "perceptual aliasing" and the "curse of dimensionality" are perhaps more important than the details of the underlying learning algorithm itself. In other words, the structure in which learning takes place is as critical as the way the learning actually occurs. Among the few projects that pull these two research strands together, Klopf [Klopf93] developed a model that accounts for a wide range of results from classical and instrumental conditioning, and Booker [Booker88] and Maes [Maes90b] have integrated learning into more complete action selection algorithms. But much remains to be done.

In this paper, our contribution is to show how certain types of associative learning may be integrated into the action selection architecture previously proposed by Blumberg [Blumberg94, Blumberg95]. The underlying learning algorithm we use is Sutton and Barto's Temporal Difference model [Sutton90], which builds and modifies portions of the behavior network in our ethologically inspired system. Furthermore, this algorithm sets several of the fundamental variables associated with our system, thus addressing some of the previous criticisms about proliferating free parameters. In particular, our system will:

- learn that an existing behavior can lead to the fulfillment of some previously-unassociated motivational goal when performed in a novel context (i.e., in more ethological language, the system will learn new appetitive behaviors).
- learn the context in which an existing behavior will directly fulfill a different motivational goal (i.e. learn new releasing mechanisms to be associated with a consummatory behavior).

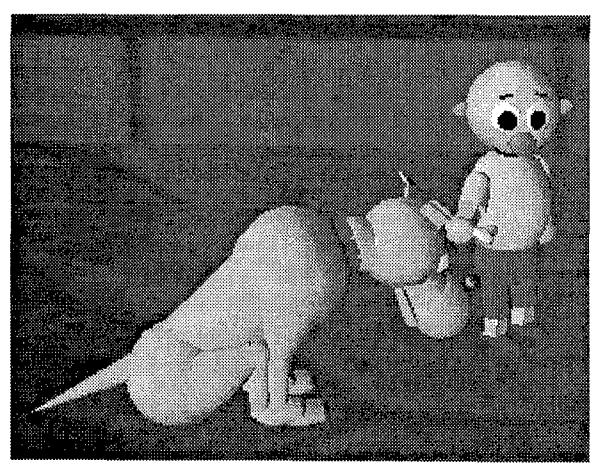


Figure 1: Silas T. Dog and his faithful teacher Dr. J.T. Puppet

* Apologies to Barbara Woodhouse [Woodhouse82].

- learn new releasing mechanisms that are found to be useful predictors of the future state of the world (i.e. show classical conditioning).

This approach to action selection combined with learning has been implemented and incorporated into the architecture for "Silas T. Dog," an autonomous, animated virtual dog featured in the ALIVE interactive virtual reality project [Maes95]. It has been successfully used to teach this old dog new tricks. By giving Silas the ability to learn, we hope to demonstrate the scientific feasibility of the coherent ethology-based model of animal behavior described in this paper. But this work also shows the usefulness of adding adaptability and user-trainability to autonomous VR agents intended to interact with people in an engaging and convincing manner. In this paper, we first discuss the important lessons from ethological research that have inspired the design of this learning system (Section 2), and then describe the basic action selection system underlying Silas's behavior (in Section 3). We proceed to the new learning mechanisms in Section 4, and indicate their impact on behavior in Section 5. Finally, we discuss the implications and uses of this approach in Section 6.

2 Lessons from Ethology

Like the previous work on action selection in Hamsterdam [Blumberg94], our present efforts to incorporate learning are inspired by decades of ethological research [Dickinson94, Gallistel90, Gallistel94, Gould94, Lorenz73, Sutton90, McFarland93, Plotkin93, Shettleworth94, Davey89]. Here we briefly list the main take-home messages we have received from this literature and used with Silas to date.

Learning complements evolution: Learning is a mechanism for adapting to significant spatial and temporal aspects of an animal's environment that vary predictably, but at a rate faster than that to which evolution can adjust (e.g. predictable changes occurring within a generation). The most adaptive course in this case is to evolve mechanisms that facilitate learning of these rapidly-varying features. Thus, evolution determines much of what can be learned and the manner in which it is learned. While this is sometimes viewed as imposing negative "constraints on learning," the innate structure in which learning occurs most often has the effect of dramatically simplifying the learning process [Gallistel90, Gallistel94, Gould94, Lorenz73, Miller90, McFarland93, Plotkin93, Shettleworth94, Todd91].

Motivations and goals drive learning: Animals learn things that facilitate the achievement of biologically significant goals. Thus, motivations for those goals drive learning. It is far easier to train a hungry animal using food as a reward than a satiated one [Gould94, Lorenz73, McFarland93]. The components of the lesson to be learned--that is, what things or actions facilitate a particular goal--are usually, but not always, contiguous in some sense with the satisfaction of the goal. For instance when a dog gets a cookie, the thing to learn is what behavior it performed just before getting that cookie.[1] (Taste-aversion learning occurring over a much longer time-period is an important example of non-temporally contiguous learning that we also wish to model.)

Animals learn new appetitive behaviors: Operant and instrumental learning can be viewed as an animal's discovery of new appetitive behaviors--that is, new behaviors that bring them closer to attaining some goal [Lorenz73]. Specifically, operant conditioning occurs when an animal finds that an existing behavior (one that was typically performed in another context for another purpose) performed in a new stimulus situation will reliably lead to a state of the world in which performance of a consummatory behavior will successfully satisfy one or more motivational variables. For example, a dog that learns to roll over on command in order to get a cookie has added the "roll over in the context of the spoken word ROLL" behavior to its appetitive behaviors for feeding.

Animals learn the value of stimuli and behaviors: Stimuli that identify motivationally significant contexts, and appetitive behaviors that lead to goal-satisfying contexts, both have learnable values associated with them. This is an underlying assumption of many associative models of classical or operant conditioning [Sutton90, Klopf93, Montague94]. It is also a necessary assumption for integrating learning into an ethologically inspired model, such as that used in Hamsterdam, in which external and internal factors are expressed as real-valued quantities.

Animals learn more than associations: It is not enough simply to learn that some action X is a useful appetitive behavior. It is also useful to learn an expected time course for X, that is, how long to engage in action X before concluding that reinforcement is not forthcoming and that another behavior may be more successful. Gallistel [Gallistel90,94] presents evidence that in certain cases animals can learn the information relevant to this kind of time course knowledge, including the number of events, the time of their occurrence and the interval between them. Within an associative model, such as the one we develop here, it is also necessary to augment the system's knowledge with such additional statistics in order to allow learning of a behavior's expected time course.

---

1 Credit assignment is a hard problem for both animals and machines. Many of the innate mechanisms which facilitate learning in animals do so by solving, in effect, the credit assignment problem. This is done by either making the animal particularly sensitive to the relevant stimuli, or predisposed to learn the lesson when it is easiest to do so (e.g. song learning) [Gould94, McFarland93, Plotkin93]. With respect to animal training, trainers stress the importance of giving reinforcement within 2-4 seconds of the performance of a desired behavior and it is very difficult to train animals to perform sequences of actions, except by chaining together individually trained actions [Rogerson92, Lorenz94]. However, by breaking a sequence of desired actions into individually trained steps, the trainers are solving the credit assignment problem for the animal. Learning and training are flipsides of the same coin: learning tries to make sense of the world and put it into as simple a framework as possible, and training tries to make the world as simple as possible for the learning mechanisms to grasp.

3 Computational Model

In this section, we summarize our ethologically inspired computational model for action selection and learning in autonomous animated creatures. The underlying architecture for motor control and action selection is described in more detail in [Blumberg94, Blumberg95]. Here we will only describe enough of the underlying architecture to make it clear how learning is integrated into this system.

3.1 Overview

The basic structure of a creature in our system consists of the three basic parts (Geometry, Motor Skills and Behavior System) with two layers of abstraction between these parts (Controller, and Degrees of Freedom).The Geometry provides the shapes and transforms that are manipulated over time to produce the animated motion. The Motor Skills (e.g. "walking," "wagging tail") provide atomic motion elements that manipulate the geometry in order to produce coordinated motion. Motor Skills have no knowledge of the environment or state of the creature, other than that needed to execute their skill. Above these Motor Skills sits the Behavior System, which is responsible for deciding what to do, given a creature's goals and sensory input. The Behavior System triggers the correct Motor Skills to achieve the current task or goal. Between these three parts are two layers of insulation, the Controller and the Degrees of Freedom (DOFs), which make this architecture generalizable and extensible. The DOFs, Controller, and Motor Skills together constitute what we call the Motor System.

A key feature of the Motor System is that it allows multiple behaviors to express their preferences for motor actions simultaneously. It is structured in such a way that Motor Skills that are complementary may run concurrently (e.g. walking and wagging the tail), but actions that are incompatible are prevented from doing so. Furthermore, the motor system recognizes 3 different imperative forms for commands: primary commands (i.e. do the action if the system is able to), secondary commands (i.e. do it if no other action objects) and meta-commands (i.e. do it this way). This allows multiple behaviors to express their preferences for motor actions.

There are at least three types of sensing available to autonomous creatures in our system:

- Real-world sensing using input from physical sensors.
- "Direct" sensing via direct interrogation of other virtual creatures and objects.
- "Synthetic vision" using computer vision techniques to extract useful information from an image rendered from the creature's viewpoint. This is used for low-level navigation and obstacle avoidance, based on ideas from Horswill [Horswill93], and [Reynolds87].

Most of the learning in our system relies on "direct" sensing, in which objects in Silas's world make information about their state available to Silas and other objects. For example, in the ALIVE interactive virtual reality system [Maes96], a user's gestures in the "real world" are converted into state variables associated with the user's proxy "creature" in Silas's world. Via direct sensing Silas can "sense" the state of this proxy creature, and thus react to the gestures of the user in the real world.

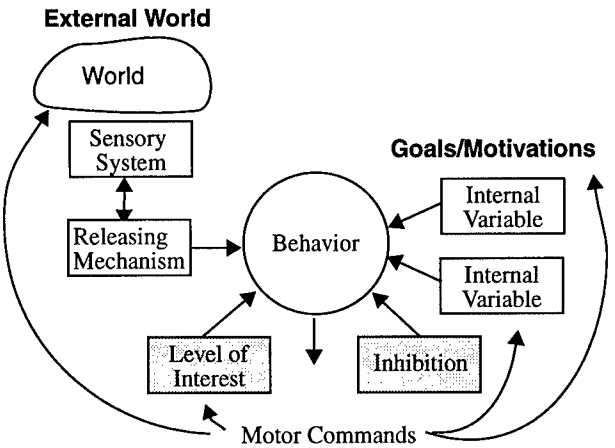


Figure 2: The purpose of a Behavior is to evaluate its relevance given external stimulus and internal motivations, and if appropriate issue motor commands. Releasing Mechanisms act as filters which identify significant objects or events from sensory input, and output a value which represents the strength of the sensory input. Motivations or goals are represented via Internal Variables which output values which represent their strength. A Behavior combines the values of the Releasing Mechanisms and Internal Variables on which it depends to yield its value before Level of Interest and Inhibition from other Behaviors. Level of Interest is used to model boredom or behavior-specific fatigue. Behaviors compete, via mutual inhibition, with other Behaviors for control of the creature.

The purpose of the Behavior System is to send the "right" set of control signals to the motor system at every time-step. The "right set" is determined by weighing the current competing goals of the creature, assessing the state of the environment, and choosing the behavior (and its associated set of actions) that best satisfies some of the creature's goals at this instant in time. More generally, the Behavior System coordinates the creature's use of its available high-level behaviors in a potentially unpredictable environment.

We will now turn to a discussion of the major components of the Behavior System and briefly describe the role of each component.

3.2 Behaviors

The Behavior System is a distributed network of self-interested, goal-directed entities called Behaviors. The granularity of a Behavior's goal can range from very general (e.g. "reduce hunger") to very specific (e.g. "chew food"). The major components of an individual Behavior are shown in Figure 2.

Each Behavior is responsible for assessing its own current relevance or value, given the present state of internal and external factors. On the basis of this assessed value, each Behavior competes for control of the creature. The current value of a Behavior may be high because it satisfies an important need of the creature (as indicated by high values of the associated Internal Variables--see section 3.4), or because its goal is easily achievable given the current state of the environment (as indicated by high values of the associated Releasing Mechanisms--see Section 3.3).

Behaviors influence the system in several ways: by issuing motor commands which change the creature's relation-

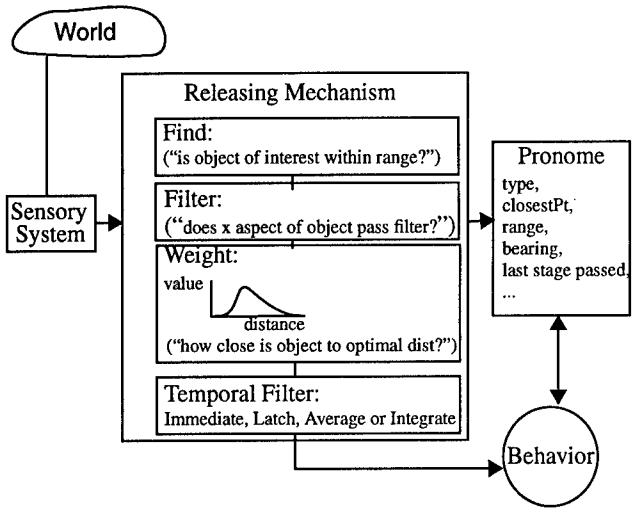


Figure 3: Releasing Mechanisms identify significant objects or events from sensory input and output a value which represents its strength. By varying the allowed maximum for a given Releasing Mechanism, a Behavior can be made more or less sensitive to the presence of the relevant stimuli. A Releasing Mechanism also fills in a data structure called a Pronome [Minsky86]. The Pronome can be thought of as representing the current focus of attention, which can be shared across behaviors.

ship to its environment, by modifying the value of Internal Variables, by inhibiting other Behaviors, or by issuing suggestions which influence the motor commands issued by other Behaviors.

Behaviors are distinguished from Motor Skills in two ways. For example, Behaviors are goal-directed whereas Motor Skills are not. For example, "Walking" is a Motor Skill whereas "Moving toward object of interest" is a Behavior which invokes the "Walking" Motor Skill to accomplish its goal. Second, Motor Skills do not decide when to become active, but rather rely on another entity to invoke them when appropriate. Motor Skills are similar to the Ethological idea of "Fixed Action Patterns" in that once initiated they do not require sensory input (other than proprioceptive input) and they have a specific time course [Lorenz76]. For the purposes of learning, the set of Motor Skills is assumed to be fixed. Thus, Motor Skill learning is an important type of learning which we do not address in this system at present. However, see [Giszter94] for an interesting example of motor learning which uses Maes' algorithm.

3.3 Releasing Mechanisms

Objects called Releasing Mechanisms (see Figure 3) filter the creature's sensory input to identify objects and events whose presence helps determine the current appropriateness of each possible Behavior. Releasing Mechanisms output a continuous value that typically depends on:

- the presence of the stimulus object or event (e.g. a person is nearby).
- more specific features associated with the stimulus (e.g. the person's hand is down and extended).
- the distance to the stimulus relative to some ideal distance (e.g. the hand is one foot away, and I like hands that are two feet away).

By representing the output of a Releasing Mechanism as a continuous quantity, this value may be easily combined with the strength of motivations (from Internal Variables) that are also represented as continuous values. This combination in turn allows the creature to display the kind of trade-off behavior one finds in nature where a weak stimulus (e.g. week-old pizza) but a strong motivation (e.g. very hungry) may result in the same behavior as a strong stimulus (e.g. chocolate cake) but weak motivation (e.g. full stomach).

Releasing Mechanisms return a value between some minimum and maximum, with the maximum value occurring when the Releasing Mechanism's object of interest appears at some optimal distance from the creature. But how can we set this maximum value for a given Releasing Mechanism? The value of the Releasing Mechanism should reflect the usefulness of its associated Behavior to the creature, and hence its maximum should be equal to the time-discounted usefulness of its object of interest (which the Behavior will exploit). For example, for a food-sensitive Releasing Mechanism, its maximum value should be equivalent to the time-discounted value of the food source to the creature. We use the time-discounted value to reflect the fact that it will take a finite period of time to consume the food. Thus, with everything else being equal, a food source that provides more energy per time-tick will be favored over one which provides less energy per tick, even if the total amount of energy from the two food sources is the same. This perspective is central to our approach to learning.

Ethologists generally view Releasing Mechanisms as filtering external stimuli. We generalize this and allow Releasing Mechanisms to be inward looking as well, so for example, a Releasing Mechanism might be sensitive to the performance of a given behavior. This latter type of Releasing Mechanism is called a Proprioceptive Releasing Mechanism (PRM) to distinguish it from External Releasing Mechanisms (ERM) which are outward looking.

3.4 Internal Variables

Internal Variables are used to model internal state such as level of hunger or thirst. Like Releasing Mechanisms, Internal Variables are each expressed as a continuous value. This value can change over time based on autonomous increase and damping rates. In addition, certain Behaviors, such as "eating," modify the value of an Internal Variable as a result of their activity. Ethologically speaking, these are "consummatory" behaviors, which when active have the effect of reducing the further motivation to perform that behavior. When a consummatory Behavior is active, it reduces the value of its associated motivational variable by an amount equal to some gain multiplied by the Behavior's value. Thus, the higher the Behavior's value, the greater the effect on the motivational Internal Variable.

Much of the learning that occurs in nature centers around the discovery either of novel appetitive behaviors (i.e. behaviors that will lead to a stimulus situation in which a consummatory behavior is appropriate), or of stimulus situations themselves in which a consummatory behavior should become active. More generally, one could say that the motivations that these consummatory behaviors satisfy drive much of the learning in animals. We adopt just such a perspective in our model: motivational Internal Variables are viewed as entities that actively drive the attempt to discover

new strategies (i.e. appetitive behaviors) that insure their satisfaction.

Based on this perspective, we can use the change in the value of a motivational variable due to the activity of a consummatory behavior as a feedback (or reinforcement) signal for the learning process. As we stated earlier in this section, changes in motivational Internal Variables are proportional to the value of the Behavior that changes them, which is in turn proportional to the value of the Releasing Mechanism that triggers it and the current level of motivation reflected in the Internal Variable value. Thus, more succinctly, motivational change-based reinforcement is proportional to both the quality of the stimulus (i.e., the value of the underlying Releasing Mechanism) and the level of motivation (i.e., the value of the changing Internal Variable).

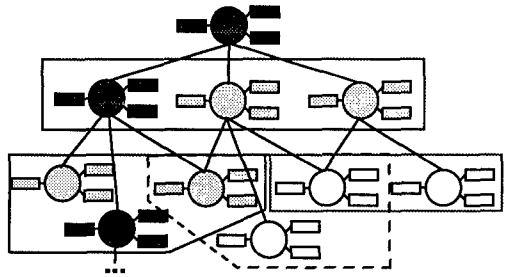


Figure 4:Behaviors are organized into groups of mutually inhibiting behaviors called Behavior Groups. Behavior Groups in turn are organized in a loose hierarchical fashion. Behavior Groups at the upper levels of the hierarchy contain general behaviors (e.g. "engage-in-feeding") which are largely driven by motivational considerations, whereas lower levels contain more specific behaviors (e.g. "pounce" or "chew") which are driven largely by immediate sensory input. The algorithm's arbitration mechanism insures that only one Behavior in a given Behavior Group will have a non-zero value after inhibition. This Behavior is then active, and may either issue primary motor commands, or activate the Behavior Group which contains its children behaviors (e.g. "search-for-food", "sniff", "chew" might be the children of "engage-in-feeding"). The dark gray nodes represent the path of active Behaviors on a given tick. Losing Behaviors in a given Behavior Group may nonetheless influence the resulting actions of the creature by issuing either secondary or meta-commands.

3.5 Behavior Groups

Behaviors are organized into mutually inhibiting groups called, simply, Behavior Groups (similar to Minsky's cross-exclusion groups [Minsky86]). These are illustrated in Figure 4. While we find a loose hierarchical structure useful this is not a requirement (i.e. all the Behaviors can be in a single Behavior Group). Behavior Groups are important because they localize the interaction among Behaviors which in turn facilitates adding new Behaviors.

Typically, the Behavior Group at the top of the system is composed of Behaviors that, in turn, have their own subsumed Behavior Groups specialized for addressing a particular need of the creature (e.g. feeding or mating). These subgroups very often have the same structure, consisting of a consummatory behavior and one or more appetitive behaviors. The different appetitive behaviors represent alternative strategies for bringing the creature into a situation in which the consummatory behavior may be used. Associated with each of the appetitive behaviors are one or more Releasing Mechanisms that signal the appropriateness of this behavior given the current environment (i.e., the likelihood that this appetitive behavior will lead to a situation in which the consummatory behavior may become active).

Learning a new appetitive behavior is accomplished by adding this Behavior to an existing Behavior Group. For example, consider a dog learning that sitting when her owner says "Sit" will often lead to a reward. In this case, the "sitting" Behavior together with a Releasing Mechanism that fires on the verbal command "Sit" are both copied into the Behavior Group associated with feeding, and thus performing this trick becomes one more strategy that the dog can use for getting food.

3.6 Inhibition and Level of Interest

Any creature has only limited resources to apply to satisfying its needs (e.g. it can only walk in one direction at a time), and thus there needs to be some mechanism to arbitrate among the competing Behaviors within a Behavior Group, since they cannot all be in charge at once. Moreover, once a creature is committed to satisfying a particular goal, it makes sense for it to continue pursuing that goal unless something significantly more important comes along.

We follow Minsky [Minsky86] and Ludlow [Ludlow76,80] in relying on a model of mutual inhibition both to arbitrate among competing behaviors in a Behavior Group and to provide control for behavioral persistence. Mutual inhibition usually (see [Blumberg94,Blumberg95]) leads to what Minsky terms the "avalanche effect," in which even weakly superior Behaviors can quickly come to dominate all the others. Ludlow provides two further important insights. The first is to recognize that by adjusting the inhibitory gains, one can vary the relative persistence of one Behavior versus the others. In general, the greater the gain, the more persistent the winning Behavior will be. The second insight is to associate a Level of Interest with every Behavior, indicating how "interested" the creature is in pursuing this Behavior right now. When the Behavior is active, its Level of Interest decreases, whether or not the Behavior achieved its goal. This in turn reduces the value of the Behavior regardless of its intrinsic value (i.e. its value before inhibition and Level of Interest were taken into account). Eventually, this will allow other Behaviors to become active. Level of Interest variables thus help the creature to avoid situations in which it obsesses on a single unattainable goal, performing the same currently ineffective Behavior over and over without result, and ignores less important but achievable goals.

Level of Interest also has a clear meaning in the context of an appetitive Behavior: it controls how long the creature should persist in the Behavior before giving up. For example, if the creature has learned from experience that a reward usually follows within X ticks of commencing some Behavior, then it should base the amount of time it will pursue this Behavior on this knowledge (i.e., it should accordingly adjust the amount by with the Level of Interest is decreased after each performance of the Behavior).

4 Integration of Learning

As indicated in Section 1, we want Silas to be able to learn: new contexts for goal-satisfying with existing behaviors; new releasing mechanisms for consummatory behaviors; and new releasing mechanisms with greater predictive

power that previously-used releasing mechanisms. The first two of these items are thinly veiled descriptions of what is usually referred to as operant or instrumental conditioning. In particular, learning new contexts for existing behaviors is essentially what animal training is all about (what Lorenz [Lorenz73] refers to as "conditioned action" or "conditioned appetitive behavior,". That is, the animal learns an association between an behavior and an outcome, and also learns the stimulus configuration upon which this association is contingent. The third type of learning corresponds more closely to "classical conditioning" in that there is already a built-in association between a stimulus and a response (i.e. a Releasing Mechanism and a Behavior), and what is learned are one or more novel Releasing Mechanisms that have an incremental predictive value (e.g. can predict an outcome further in the future). In the first two cases, a change in an underlying motivational variable acts as the reinforcement or feedback signal. By contrast, in the third case feedback is provided by an existing Releasing Mechanism. In the rest of this section we outline how these types of associative learning can be integrated into the action selection architecture described in Section 3.

4.1 Conceptual Overview of Learning Process

Fundamentally, the animal is trying to learn one of a number of potential associations:

- (S→O): Context S leads to outcome O, regardless of behavior
- (A→O): Behavior A leads to outcome O, regardless of context
- (S→(A→O)): Behavior A leads to outcome O, in context S

Here, outcomes mean a significant change in a motivational variable. Given these possible associations in the world, what should the learning system pay attention to and make associations with when a particular outcome occurs?

Temporal and spatial contiguity answers the first part of this question. That is, when a motivational variable changes, the learning system looks at Silas's recent behaviors and at the recently changed features of (nearby) objects of interest, to see if any of these may be associated with the outcome. Essentially, Silas says "let's check short term memory and start watching the last *N* behaviors and objects of interest to see if they continue to be associated with changes in this motivational variable from now on."

We use Sutton and Barto's temporal difference model of Pavlovian conditioning [Sutton90] to actually learn the prevailing associations. This model effectively forces alternative "explanations" (i.e., behaviors or object features) to compete to "explain" (i.e., make the best prediction of) the time-discounted changes in the value of the reinforcement variable (i.e., the outcome). Sutton and Barto's model is attractive because it is simple, yet it explains most of the results of Pavlovian conditioning experiments. It results in Silas learning a value for a particular behavior or stimulus that reflects its time-discounted association with changes in the value of some motivational variable.

In the first two types of associations, the change in the value of the underlying motivational variable (i.e., the outcome) acts as the reinforcement variable that drives learning, making simple first-order associations. In the third case (S→(A→O)), the learned value of A (in producing outcome O) acts as the reinforcement variable for learning O's association with S. This allows Silas to learn second-order associations.

The fundamental insight that allows us to integrate this learning algorithm into our existing action selection algorithm is that the "value" to be learned for a behavior or stimulus is represented by the MaxValue associated with the Releasing Mechanism sensitive to that behavior or stimulus. In order to treat behaviors as learnable "cues" in the same way as external stimuli, we use PRMs (Proprioceptive Releasing Mechanism) to signal when a given behavior is active. In all cases, then, we are learning the appropriate MaxValue associated with a given Releasing Mechanism.

When the learned MaxValues of the relevant Releasing Mechanisms get above a threshold, this means that the association is "worth remembering," and should be made a permanent part of the Behavior network. For example, in the case of a (S→(A→O)) association worth remembering, this means that in context S, signalled by one or more Releasing Mechanisms, performance of behavior A has reliably led to outcome O, which corresponds to the performance of a consummatory behavior that reduces the value of the underlying motivational variable. More concretely, in the case of sitting (A) on command (S) for food (O), the sitting behavior and the set of releasing mechanisms (e.g., an outstretched hand) that indicate it's time for Silas to do his trick will be added to the Behavior Group that is principally concerned with feeding. In addition, the Releasing Mechanisms that signal S will be added to the top-level behavior that owns the Feeding Behavior Group. This last step is necessary because S, the sitting-trick context, is now a relevant context for engaging in Feeding.

Because in this example (S→ (A→O)) represents a new appetitive strategy, it is also important to learn the appropriate time-course for the actions involved. That is, Silas should learn how long he should engage in the particular behavior of sitting, waiting for a cookie, before giving up. This information is computed as a by-product of the learning process and is used to change the Level Of Interest parameter of the behavior over time (see Section 4.2.6).

4.2 Extensions to Support Learning

To add learning to the action selection mechanisms presented in Section 3, we have developed several new structures that we now describe in turn.

4.2.1 Short-Term Memory

The Behavior System maintains FIFO memories that are essential for learning. One memory keeps track of the last *N* unique behaviors that have been active, and another keeps track of the last *N* objects of interest (currently we use *N* = 10). The active behavior on a time-tick is the leaf behavior which ultimately won during action selection. The object of interest is the pronome associated with that behavior. Note that in both cases the memory is based on unique instances and not time. This idea is taken from Killeen: "Decay of short-term memory is very slow in an undisturbed environ-

ment and dismayingly fast when other events are interpolated...Such distractors do not so much subvert attention while time elapses but rather by entering memory they move time along" [Killeen94].

Foner and Maes [Foner94] also emphasize the importance of contiguity in learning. However, because short-term memory in our system is based on unique behaviors or objects of interest, our focus of attention is not strictly equivalent to temporal or spatial contiguity. In addition, we will probably move to a system in which each motivational variable has its own memory-list of potentially significant objects or events, again stripped of their temporal information. This would then make it possible to model long-time-span learning phenomena such as taste aversion.

4.2.2 Reinforcement Variables

A Reinforcement Variable is either an Internal Variable (i.e. a motivational variable), or a Releasing Mechanism (either an ERM or a PRM) whose value is used as a feedback signal for learning. For example, hunger would be a Reinforcement Variable if the animal wished to learn new appetitive strategies to reduce its level of hunger. Alternatively, the PRM associated with the "Begging" behavior would be the Reinforcement Variable associated with learning the context in which Begging was a successful appetitive behavior for reducing hunger.

4.2.3 Discovery Groups and the Learning Equation

A Discovery Group is a simply a collection of Releasing Mechanisms (either PRMs or ERMs) for which the animal wants to discover the appropriate association with a Reinforcement Variable. The Releasing Mechanisms in a Discovery Group effectively compete to "explain" the value of the Reinforcement Variable, and as they learn their association with the Reinforcement Variable, they change their maximum value accordingly. Thus, a Releasing Mechanism that has no apparent association with the state of a Reinforcement Variable will wind up with a maximum value close to zero. By contrast, a Releasing Mechanism which has a strong association will wind up with a maximum value that approaches the time-discounted value of the Reinforcement Variable.

We now turn to the question of how Discovery Groups are formed, that is, how creatures "decide" what to watch as potential sources of important associations. The first point to be made is that it is there is no "centralized" learning mechanism. Rather, each motivational variable is responsible for learning its own set of appetitive behaviors and their respective contexts using the mechanisms described in the previous sections. This means that each motivational variable has its own collection of Discovery Groups for which it is the Reinforcement Variable. Thus, exogenous changes (i.e. changes directly due to the performance of a behavior) in the motivational variable will serve as the feedback for the learning process in these Discovery Groups. This process is summarized in Figure 5.

The second point is that significant changes in the level of the motivational variable initiate the process of identifying candidates (i.e. ERMs or PRMs) to add to the Discovery

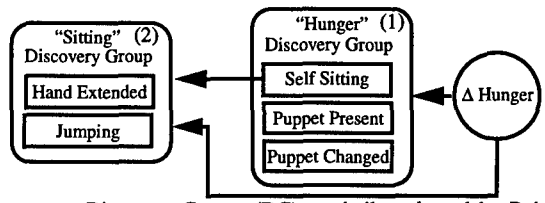


Figure 5: Discovery Groups (DG) are built and used by Reinforcement Variables (e.g. motivational variables) to explain significant changes in their value. For example, when hunger undergoes a significant change, it looks at short-term memory for the most recent behaviors performed and objects of interest encountered and adds the appropriate RMs (e.g. "Self-Sitting") to its primary DG (1). Within a DG, the Barto-Sutton TD learning algorithm is used to discover the appropriate MaxValue for each RM. In order to learn the appropriate context for a behavior which appears to be important, a subsidiary DiscoveryGroup (2) may be constructed which has both the motivational variable and the proprioceptive RM for that behavior as its reinforcement variables. It will be populated with RMs which are sensitive to those features (e.g. "Hand-Extended") of the recently encountered objects of interest which appear to change shortly before the motivational variable undergoes its change.

Groups. The fundamental assumption is one of contiguity: i.e., that one or more of the creature's recent behaviors and/or one or more of the features associated with the creature's recent objects of interest may have a causal or predictive association with the change in the motivational variable. In the current system, this means choosing candidates from the contents of short-term memory.

Discovery Groups and Reinforcement Variables are intended to be an abstraction of the temporal difference associative models proposed by researchers such as Klopf [Klopf93] or Sutton and Barto [Sutton90] to explain the results of classical conditioning. In their context, the Reinforcement Variable corresponds to the Unconditional Stimulus (UCS), and the Discovery Group corresponds to the collection of Conditional Stimuli (CS) that are competing to "explain" the value of the UCS.

We follow the Sutton and Barto TD learning model in this paper. Thus, within a Discovery Group we use the following equation to learn the association between a Reinforcement Variable and one or more Releasing Mechanisms:

$$\Delta V_{it} = \beta\left(\left[\lambda_t + \gamma\left[\sum_j V_{j(t-1)} \cdot A_{jt}\right] - \left[\sum_j V_{j(t-1)} \cdot A_{j(t-1)}\right]\right]\alpha\bar{A}_{it}\right)$$

Where:

V_{it} = MaxValue of RM i at time t

λ_t = Value of reinforcement variable at time t

$\bar{A}_{it}$ = Trace for RM i at time t

A_{it} = 1 if RM i passed find/filter and weight > .90, 0 otherwise

β = Learning Rate

γ = Temporal Discount Rate

α = Trace Rate

δ = Trace Decay Rate

Where the trace is defined as:

$$\bar{A}_{it} = \bar{A}_{i(t-1)} + \delta(A_{i(t-1)} - \bar{A}_{i(t-1)})$$

The first equation specifies how the MaxValue associated with a given RM in the Discovery Group will change in response to the value of the Reinforcement Variable. The amount it will change is proportional to an error defined to be the feedback actually received plus the discounted prediction of future feedback by all the Releasing Mechanisms in the Discovery Group less the prediction of feedback by all the Releasing Mechanisms on the previous time step. In

other words it changes by an amount proportional to the amount of actual and predicted feedback unexplained by the set of Releasing Mechanisms in the Discovery Group. In effect, they are competing for value, where the value is the time-discounted value of feedback.

The second equation is used to specify a "stimulus" or memory trace for a given Releasing Mechanism. It basically does temporal averaging of the state of the Releasing Mechanism, where A is either 1 or 0 (i.e. active or inactive) and A bar ranges between 0 and 1. This term is used by Sutton and Barto to explain temporally discontinuous associations between a CS and a UCS (e.g. when the CS turns off prior to the start of the UCS).

4.2.4 A Variable Learning Rate

A variable learning rate is a necessary addition to the learning equation in order to explain two results found in animal learning. First, prior exposure to a CS without a correlated UCS delays learning a later association between the two. Second, it is well known that partial reinforcement, while lengthening training times, also lengthens the time to extinction. Neither of these effects would be predicted by the standard fixed-rate learning equation. Gallistel [Gallistel90] and Cheng [Cheng95] base their arguments against associative models in part on just these reasons.

To rescue associative learning, we propose that the learning rate should be proportional to some measure of the observed reliability of receiving, or not receiving, feedback. If the reliability is high -- e.g., a behavior is always rewarded, or always not rewarded -- then it makes sense to have a high learning rate, i.e. to pay attention to each observation. On the other hand, if the observed reliability is low -- e.g., a behavior is sometimes rewarded, and sometimes not -- then the learning rate should be low, reflecting the uncertainty associated with any one observation.

To implement this intuition, we base the learning rate on a moving average of what we call the "Reliability Contrast" which is defined to be:

$$RC = \frac{\#(\text{active \& feedback})}{\#(\text{active})} - \frac{\#(\text{active \& no feedback})}{\#(\text{active})}$$

In other words, the RC is the difference between the observed probability of receiving feedback at least once while the behavior is active and the observed probability of receiving no feedback while the behavior is active. The RC ranges from reliably-unreliable (-1.0)—e.g. you will reliably not receive feedback to reliably-reliable (1.0)—e.g you will reliably receive feedback. The equation for the learning rate is then:

$$\beta = \beta_{min} + \text{fabs}(RC)(1 - \beta_{min})$$

This variable learning rate allows us to accommodate one of the two weaknesses of the learning equation, namely the partial reinforcement effect. It will also accommodate the phenomena of rapid re-acquisition after extinction because in extinction the learning rate is high, and so it responds to the change rapidly. Both effects are shown in Figure 6. It does not, however, accommodate the phenomena of prior exposure reducing the learning rate. We also do not yet model the transition from a learned response to essentially a habit (i.e. when the learning rate decays to 0 and learning stops).

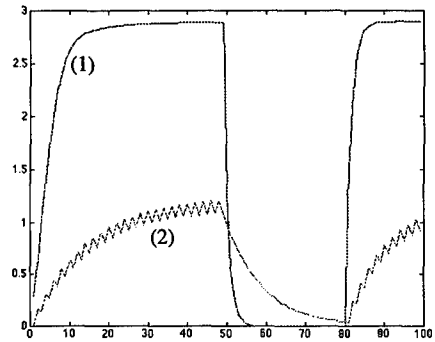


Figure 6: The graph shows the effect of a variable learning rate on the time-course in the associative strength of a stimulus under acquisition (t<50), extinction (50<t<80), and reacquisition (t>80) in 2 feedback ratio scenarios ((1) = rewarded each time, and (2) = rewarded every other time). The learning rate is calculated in each case on a moving average of the reliability contrast. Since the reliability contrast is lower in the case of the variable reward scenario (2), its learning rate is lower and thus its decay in extinction is more gradual.

4.2.5 Learning the Level of Interest

As mentioned earlier, in a world of uncertain rewards an animal needs to learn not just new appetitive behaviors, but also how long to engage in a given appetitive behavior before giving up and trying something else. We use a very simple approach to this problem: as part of the learning process, the Behavior incorporates its expected time to feedback and the associated variance. The Level of Interest for the Behavior is then held constant (and high) over the expected time to reward, and from that point on is allowed to decay linearly to zero over a period of time.

5 Implementation & Results

The algorithm described in the previous sections has been implemented as part of the Hamsterdam toolkit for building and controlling autonomous animated creatures. We have developed several creatures using this toolkit, including the ALIVE [Maes96] project's previously-mentioned virtual dog, Silas, a Puppet, and a Hamster [Blumberg95, Blumberg94]. Here we describe some of the initial results we have obtained for Silas and the hamster learning new positive and negative environmental associations.

Silas has 24 dog-specific Motor Skills and responds to 70 motor commands. In his current simplified form for testing learning, Silas initially has a Feeding Behavior Group which is empty except for a consummatory "eat" behavior--that is, this Behavior Group has no appetitive behaviors. Silas also has an exploratory Behavior Group that causes him to approach the user and randomly engage in a variety of different behaviors (e.g. sitting, begging, lying-down etc...).

We have used the learning algorithm described in the previous section to successfully train Silas to respond to gestures made by another autonomous creature, the Puppet (see Figure 1). The moment the Puppet detects that Silas is performing a desired behavior[2] (e.g. sitting), he performs the training gesture (e.g. hand-extended-down), and 10 ticks later (approximately 1/2 second), gives Silas a dog bone. Silas successfully learns that sitting when the Puppet's hand is extended is a useful appetitive strategy for getting food, and thus adds the "Sit" behavior and its associated RM for

"hand-extended" to its "Feeding" system.

As the next step in teaching Silas to sit, the Puppet (an unexperienced dog-trainer) changes his training gesture to be the combination of "hand extended" and "jumping." However, as a result of blocking, Silas attributes little value to the "jumping" component of the combined gesture, because it has no incremental predictive value beyond the "hand extended" component. But when the Puppet changes his training paradigm yet again and begins to jump several ticks before he extends his hand, Silas rapidly learns the new significance of "jumping". This is shown in Figure 7.

This experiment shows that the learning system described here can produce some of the phenomena associated with classical conditioning (not altogether surprising given our learning equation). Silas also learns the expected time to reward and adjusts the Level of Interest associated with "sitting" in this context accordingly, as discussed in Section 4.2.5.

In another experiment, we have a Hamster which is engaged in foraging in an open pen. The floor of the pen can be "electrified" and its color can change. The experiment is set up so that the floor-color will go from off to green to yellow to red. When the color is red, a shock is delivered for up to 40 ticks as long as the Hamster remains within a certain radius of the center of the pen. If the Hamster leaves the critical area, the shock is immediately turned off. The Hamster has a built-in response to shock: it will randomly choose to either "freeze" or to "run away." It also has a built-in association that the shock comes from the environment (i.e. when it is shocked the environment becomes its object of interest). The motivational variable is "relief from pain." Since the Hamster's speed is such that it can leave the pen in substantially less time than the shock's duration, running away should be the preferred action, and it should have an increasing tendency to run away as the floor-color progresses from green to red.

Over the course of 20 trials (i.e. 20 individual shocks), the Hamster learns that running away brings a quicker reduction in pain than does freezing, and so running away becomes the preferred option. In addition, it learns the association between the time course of the colors and being shocked and begins to flee when it senses the green light, or the yellow light at the latest. As a result, it successfully learns to avoid getting shocked.

We have also performed an experiment similar to that described by Montague [Montague94], in which our hamster learns to preferentially approach one food source over another based on the relative value of the alternative food sources. However, this needs to be combined with a motivationally-based exploration strategy so that the Hamster will periodically test the less-preferred source to see if its value has changed. These results, along with those for Silas's training, are still early and informal, but together they illustrate the power of the approach taken in this paper.

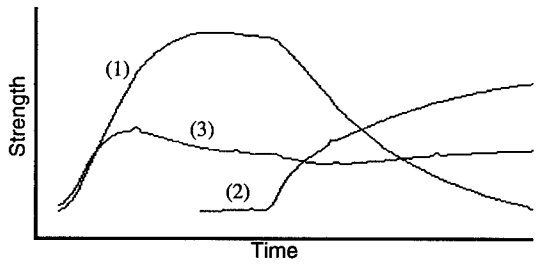


Figure 7: This shows the change in MaxValue for the "extendedRight" ERM (1), "jumping" ERM (2) and the "Sit" PRM (3) as a result of being associated with food during training. Initially, the Puppet uses a single gesture ("extendedRight"), but midway through training begins to simultaneously combine another gesture, "jumping". As a result of blocking, little associative strength is allocated to the "jumping" PRM (flat portion of 2) until the Puppet changes its training paradigm again and "jumping" precedes "extendedRight" by 3 ticks. As a result, "jumping" rapidly gains strength at the expense of "extendedRight".

6 Areas for future work

Using the learning algorithm presented here we plan on demonstrating further learning effects from the ethological and psychological literature. For example, we want to demonstrate different phenomena including taste aversion, outcome devaluation, habit formation, learned helplessness and superstitious behavior. From the ethological literature, we want to demonstrate socially-facilitated learning [Gould94, Shettleworth95] and integrate an exploration vs. exploitation strategy which is under the influence of motivational variables.We would also like to be able to demonstrate chaining to produce behavioral sequences (e.g. as in a circus routine).

Learning new motor skills (or combinations of motor skills) represents a major area of behavioral innovation which is not addressed here and thus represents a limitation in the current work. However, the architecture of the Motor System would make it straightforward to add this type of learning.

7 Conclusion

Our contribution in this paper is to show how certain types of learning may be integrated into the animat architecture previously proposed by Blumberg [Blumberg94, Blumberg95]. The underlying algorithm is Sutton and Barto's Temporal Difference model, but we show how it may be used and interpreted within the context of our ethologically inspired model to build and modify portions of the behavior network, and to set several of the fundamental parameters associated with our system.

It is not the intent of the authors to "push" one learning algorithm vs. another, but rather to stimulate thought on how best to integrate learning techniques such as TD learning into ethologically inspired multi-goal action selection architectures. This is done particularly with an eye towards building autonomous creatures such as "Silas" which must interact with, and learn from, users in natural ways.

2 Typically a dog trainer would issue the command "Sit" and then manipulate the dog into a sitting position [Rogerson92,Lorenz94]. Hence, the command precedes performance. To accomplish this in our system the Puppet considers the Dog to be performing a behavior the moment the behavior becomes active. However, the Dog (i.e. its PRMs) does not consider itself to be performing a behavior until both the behavior and its underlying motor skills are active. The effect is that from the Dog's perspective the command precedes its perception of its own performance.

We also hope that we have given the reader a sense of how ideas from ethology may be profitably incorporated into architectures for autonomous animated creatures.

References

Blumberg, B. (1994). Action-Selection in Hamsterdam: Lessons from Ethology. In: *From Animals To Animats, Proceedings of the Third International Conference on the Simulation of Adaptive Behavior*, Cliff, D., P. Husbands, J.A. Meyer, and S.W. Wilson, eds. MIT Press, Cambridge Ma.

Blumberg, B. and T. Galyean (1995). Multi-level Direction of Autonomous Creatures for Real-Time Virtual Environments. In: *Proceedings of SIGGRAPH 95.*

Booker L. (1988). Classifier Systems that Learn Internal World Models. *Machine Learning Journal*, Volume 1, Number 2,3.

Brooks, R. (1986). A Robust Layered Control System for a Mobile Robot. *IEEE Journal of Robotics and Automation* RA-2.

Cheng, P. and K.J. Holyoak (1995). Complex Adaptive Systems as Intuitive Statisticians: Causality, Contingency, and Prediction. In: *Comparative Approaches to Cognitive Science,* Roitblat, H.L. and J.A. Meyer, eds. MIT Press, Cambridge Ma.

Davey G. (1989). *Ecological Learning Theory.* Routledge Inc., London.

Dickinson, A. (1994). Instrumental Conditioning. In: *Animal Learning and Cognition,* Mackintosh, N.J. ed. Academic Press, San Diego.

Foner, L.N. and P. Maes (1994). Paying Attention to What's Important: Using Focus of Attention to Improve Unsupervised Learning. In: *From Animals To Animats, Proceedings of the Third International Conference on the Simulation of Adaptive Behavior*, Cliff, D., P. Husbands, J.A. Meyer, and S.W. Wilson, eds. MIT Press, Cambridge Ma.

Gallistel, C.R. (1990). *The Organization of Learning.* MIT Press, Cambridge Ma.

Gallistel, C.R. (1994). Space and Time. In: *Animal Learning and Cognition.* Mackintosh, N.J. ed. Academic Press, San Diego.

Giszter, S. (1994). Reinforcement Tuning of Action Synthesis and Selection in a Virtual Frog. In: *From Animals To Animats, Proceedings of the Third International Conference on the Simulation of Adaptive Behavior*, Cliff, D., P. Husbands, J.A. Meyer, and S.W. Wilson, eds. MIT Press, Cambridge Ma.

Gould, J.L. and C.G. Gould (1994). *The Animal Mind.* Scientific American Library, New York.

Horswill, I. (1993). A Simple, Cheap, and Robust Visual Navigation System. In: *Second International Conference on the Simulation of Adaptive Behavior.* Honolulu, HI. MIT Press

Kaelbling, L. (1992). *Learning in Embedded Systems*, MIT Press, Cambridge Ma.

Killeen, P.R. (1994). Mathematical principles of reinforcement. *Behavioral and Brain Sciences*, 17,105-172.

Klopf, A.H., J.S. Morgan, and S.E. Weaver (1993). A Hierarchical Network of Control Systems that Learn: Modeling Nervous System Function During Classical and Instrumental Conditioning. *Adaptive Behavior,* Vol. 1, No. 3.

Lin, L.J., and T.M. Mitchell (1992). *Memory Approaches to Reinforcement Learning in Non-Markovian Domains,* CMU-CS-92-138, Dept. of Computer Science, Carnegie-Mellon University.

Lorenz, K. (1973). *Foundations of Ethology.* Springer-Verlag, New York.

Lorenz, K. (1994). *Man Meets Dog.* Kodansha America. New York.

Ludlow, A. (1976). *The Behavior of a Model Animal. Behavior,* Vol. 58.

Ludlow, A. (1980). The Evolution and Simulation of a Decision Maker. In: *Analysis of Motivational Processes,* Halliday F.T. &T. eds. Academic Press, London.

Maes, P. (1990a). Situated Agents Can Have Goals. *Journal of Robotics and Autonomous Systems* 6(1&2).

Maes P. & R. Brooks (1990b). Learning to Coordinate Behaviors. In: *Proceedings of AAAI-90.*

Maes, P. (1994). Modeling Adaptive Autonomous Agents. *Artificial Life*, Vol. 1, Numbers 1&2.

Maes, P., T. Darrell, B. Blumberg, and A. Pentland (1996). The ALIVE System: Wireless, Full-Body Interaction with Autonomous Agents, (to appear in the ACM Special Issue on Multimedia and Multisensory Virtual Worlds, spring 1996)

Mahadevan S. and J. Connell (1991). Automatic Programming of Behavior-Based Robots using Reinforcement Learning. In: *Proceedings of the Ninth National Conference on Artificial Intelligence.* MIT Press, Cambridge Ma.

McFarland, D. (1993). *Animal Behavior.* Longman Scientific and Technical, Harlow UK.

Miller, G.F., and P.M. Todd (1990). Exploring adaptive agency I: Theory and methods for simulating the evolution of learning. In: *Proceedings of the 1990 Connectionist Models Summer School,* Touretzky D.S., J.L. Elman, T.J. Sejnowski, and G.E. Hinton, eds. Morgan Kaufmann, San Mateo, Ca.

Minsky, M. (1988). *The Society of Mind.* Simon & Schuster, New York.

Montague, P.R., P. Dayan, and T.J. Sejnowski (1994). Foraging in an uncertain environment using predictive Hebbian learning. In: *Advances in Neural Information Processing 6*, Cowan, J.D., Tesauro, G, and Alspector, J. eds. Morgan Kaufmann, San Mateo, Ca.

Plotkin, H.C. (1993). *Darwin Machines and the Nature of Knowledge.* Harvard University Press, Cambridge.

Reynolds, C. W. (1987). Flocks, Herds, and Schools: A Distributed Behavioral Model. In: *Proceedings of SIGGRAPH 87.*

Rogerson, J. (1992). *Training Your Dog.* Howell Book House, London.

Shettleworth, S.J. (1994). Biological Approaches to the Study of Learning. In: *Animal Learning and Cognition,* Mackintosh, N.J. ed. Academic Press, San Diego.

Sutton, R., and A.G. Barto (1990). Time-Derivative Models of Pavlovian Reinforcement. In: *Learning and Computational Neuroscience: Foundations of Adaptive Networks.* Gabriel, M. and J. Moore, eds. MIT Press, Cambridge Ma.

Sutton R. (1991). Reinforcement Learning Architectures For Animats. In: *From Animals To Animats, Proceedings of the First International Conference on the Simulation of Adaptive Behavior*, Meyer, J.A. and S.W. Wilson, eds. MIT Press, Cambridge Ma.

Todd, P.M., and Miller, G.F. (1991). Exploring adaptive agency II: Simulating the evolution of associative learning. In: *From animals to animats: Proceedings of the First International Conference on Simulation of Adaptive Behavior*, Meyer J.A., and S.W. Wilson, eds. MIT Press, Cambridge Ma.

Tu, Xiaoyuan and D. Terzopoulos (1994). Artificial Fishes: Physics, Locomotion, Perception, Behavior. In: *Proceedings of SIGGRAPH 94.*

Tyrrell T. (1993). *Computational Mechanisms for Action Selection.* Ph.D. Thesis, Centre for Cognitive Science, University of Edinburgh.

Watkins C. (1989). *Learning from Delayed Rewards.* Ph.D. Thesis, King's College, Cambridge.

Whitehead S.D. (1992). *Reinforcement Learning for the Adaptive Control of Perception and Action.* Technical Report 406, University of Rochester Computer Science Dept.

Woodhouse, B. (1982). *No bad dogs: the Woodhouse way.* Summit Books, New York.

Generalization in Instrumental Learning

Christian Balkenius
Lund University Cognitive Science
Kungshuset, Lundagård
S-222 22 LUND, Sweden
christian.balkenius@fil.lu.se

Abstract

This paper shows how a representation at multiple scales can be used for generalization in instrumental learning. The use of such representations is an efficient way to code similarities and differences within a stimulus dimension, and allows a learning system to generalize easily between various situations. A neural network based model of classical and instrumental conditioning is presented and its ability to generalize using multi-scale representations is subsequently demonstrated in a number of simulations.

1 Introduction

Generalization, put simply, is the ability to behave in a new situation in a way that has been learned in other similar situations in the past. In nature, an animal will never encounter the exact same situation twice and if learning is to be of any use, it is necessary to generalize learned behavior to new situations. Must a study of learning not also be an inquiry into generalization?

I want to suggest that this is indeed the case, and we will see that the key concept in such an investigation is that of similarity. To determine whether it is appropriate to behave in some certain way, an animal must check if the new situation is similar to other situations in a suitable way. Generalization is only possible when this similarity can be recognized in a useful way.

When generalization is studied in the animal laboratory, an animal is first taught to respond to a specific stimulus, for example a tone. Later, it is presented a new stimulus that resembles the original in some sense, for example a tone of a different pitch (Mackintosh 1974). The amount of responding to the new stimulus is used as an index of the level of generalization. Given that some level of generalization is shown, what aspect of the animal nervous system makes this possible? Is this a phenomenon at receptor level or are more central mechanisms involved? Moreover, are there any similarities in the generalization within a single stimulus-dimension, as between two pitches, and in more complex cases, as, for example, when a child mistakes a horse for a dog?

In the next section, I show how stimuli can influence a neural network in a way that supports generalization along stimulus dimensions, although the dimensions themselves need not be explicitly represented. In sections 3 and 4, these constructions will be put to work in a model of conditioning that can generalize in a natural way. A number of simulation studies are presented that show how the model can describe generalization in instrumental learning. It is shown that the ability to generalize greatly increases the learning speed in a simple grid environment.

2 Neural Representation of Similarity

The notion of similarity that will be used here is based on the idea that a stimulus gives rise to a distributed pattern of activity in the nervous system whose components serve to represent different aspects of the stimulus. Such an activity pattern will be called a *sensory schema* (Balkenius 1994) and its different parts are assumed to directly code the values on various *stimulus dimensions*, such as shape and color. This idea is consistent with the view that the nervous system uses *representation by place*, that is, activity at a certain *location* in the nervous system corresponds to some specific stimulus property (Martin 1991).

For example, a specific set of neurons may react each time a tone of a certain pitch is heard. It follows that two

sensory schemata are similar to the extent that their neural realizations physically overlap (Balkenius 1994). Representations of this kind can be called *semi-local* since they share properties with both local and distributed coding schemes (Thorpe 1995). They are distributed in the sense that many nodes are necessary to represent the stimulus, but local in the sense that the activity of each node can be given meaning independently.

In Pavlov's (1927) classical account of generalization, it was assumed that each stimulus is represented by activity in a specific region of the cerebral cortex. Stimulus situations that were similar to each other would give rise to activity in overlapping regions of the cortex. This was the origin of the idea that generalization is based on common elements (Mackintosh 1974). According to this view, every stimulus can be analyzed into a number of components that each has the same properties as a whole stimulus. For example, a red ball can be though of as the compound of one stimulus component for the color red and one for the shape round.

Returning to the example of generalization along the pitch dimension, we see that this could be accomplished if the pitch of each tone is represented as a number of components where some of them are shared with tones of similar pitch. Such a coding is easily constructed if pitch is represented by the activity over a set of detectors, each of which is tuned to a specific frequency (figure 1).

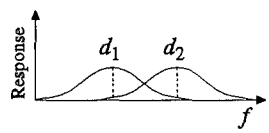


Figure 1 The response characteristics of two tuned detectors, d_1 and d_2. As a tone is moved along the frequency dimension (f), the responses of the detectors gradually change in the way specified by the tuning curve of each detector.

Since the tuning curves of the detectors overlap, each pitch will activate a whole number of detectors with strengths that depend on how well the stimulus pitch matches the preferred frequency for that detector. This classical account for generalization has a number of attractive properties (Mackintosh 1974), but without a more detailed characterization, it is impossible to say whether representations of this type can support generalization in more than the simplest cases.

For example, this coding scheme is biased toward a specific resolution in the representation of a dimension. It may be able to explain the generalization to neighbouring pitches, but it cannot explain how an animal can generalize to a tone of any pitch. Furthermore, it assumes that there exists an objective stimulus dimension to which the detector can be tuned. This may not be much of a problem for the representation of pitch, but what is the dimension underlying dog-ness or horse-ness?

To simplify the presentation below, I will continue to assume that there exists some objective stimulus dimension. In section 5, however, this requirement will be relaxed and I will describe how generalization can be viewed as a mechanism that *constructs* subjective stimulus dimensions.

To allow a fuller representation of a stimulus dimension, the notion of a scale-hierarchy is required. Each dimension must be represented by a collection of scales, each of which codes for the same value on the corresponding stimulus-dimension but with varying resolution. Representations of this type have been used for a long time in artificial vision (Witkin 1983). The different scales correspond to the visual image projected onto different frequency domains. In the coarse scales, the image is 'blurred' and does not contain any fine details. Different objects thus become more similar. In the finer scales, more details are retained, which makes it simpler to discriminate between different objects although an object may be impossible to identify if it is viewed from a slightly altered angle. One of the key ideas of the present work is to investigate if this type of *multi-scale* representations can be used more generally.

Given an initial neural activity pattern at the sensory receptors, a stimulus representation at multiple scales can be generated in one of two ways. The first is to use sets of neural detectors that are tuned to each of the relevant scales. This is illustrated in figure 2a. The second strategy uses detectors only for the finest scale and derives the coarser scales using a network lattice as shown in figure 2b. To accomplish this, the nodes on each higher level should react when one of its two nodes in the finer scale is active, that is, it should react to the *disjunction* of each of its parts in the finer scale. However, this strategy has the disadvantage that it uses a hard-wired scale-hierarchy. Such a network can only be constructed if the topology of the sensory space is known *a priori*.

3 A Neural Network Model of Conditioning

Armed with the appropriate representations for generalization we must now consider a learning model capable of using this type of representations. It is now generally acknowledged that behavior in higher animals is controlled both by associative mechanisms related to those studied within the behavioristic tradition and by higher cognitive processes. Stimulus generalization plays a crucial role in both these types of control, but for the sake of simplicity, I will only consider the first type of learning here.

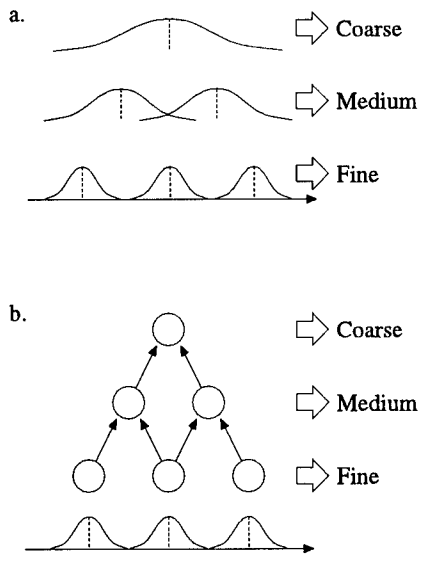


Figure 2 Two ways to derive stimulus representations at multiple scales. (a) Different sets of detectors are independently tuned to various scales. (b) The coarse representation is derived from a finer-scale representation in a feed-forward network where each node reacts to the disjunction of each of its parts at the finer level.

There are a number of requirements that a learning system must fulfill to use multi-scale representations. The first two requirements are shared with other models of instrumental learning:

1. It must be able to chain the responses required to move from the start to the goal.
2. Learning must be reversible.

The following four requirements makes sure the model can handle multi-scale representations:

3. It must operate with multiple-element stimulus representations, that is, sensory schemata.
4. The learning needs to be invariant to the size of the schemata. Learning should not be influenced by how many components are used for each schema.
5. The learning system should be able to discriminate between the different schema components. In this context, this means that it must be able to select the appropriate scale for its current problem.
6. The system must learn at different time scales. This final requirement is somewhat technical. Since the coarse representations do not necessarily change every time an action is performed, it is necessary that the system can learn even if there is a substantial time interval between changes in this representation.

Figure 3 shows an artificial neural network with the desired properties. It can also model many aspects of classical conditioning such as acquisition, extinction, conditioned inhibition, overshadowing, blocking (See Balkenius 1995 and 1996). Since my primary interest here is its ability to generalize, I will not dwell on these matters. However, it will be necessary to consider how the network models acquisition, extinction and secondary conditioning since these abilities are necessary for the generalization experiments described below. The network will be called a *reinforcement module* since its task is to calculate reinforcement, that is, signals that changes the state of the network, from changes in its input. The function of the module is related both to Gray's (1975) symmetrical model of conditioning and to Grossberg's (1987) dipole theory.

3.1 Acquisition

In the simplest form of a conditioning experiment, a conditioned stimulus, CS, is paired a number of times with an unconditioned stimulus, US, until the presentation of the CS on its own is able to produce a conditioned response, CR. The US can be either positive or negative depending on its consequences for the animal. Since the same stimuli can be used in instrumental conditioning as reward or punishment, they can alternatively be called Rew or Pun. In a typical acquisition experiment, the CS is paired with a rewarding US.

It has been shown in a number of studies that conditioning is easiest when there is a short interstimulus interval between the two stimuli (Mackintosh 1974). This is modelled in the network in the simplest possible way by demanding that the US should follow the CS by one time-step. This approximation is admittedly too crude for a more detailed modelling of interstimulus interval effects, but it will be sufficient for my present purposes.

The conditioning network consists of four nodes, x^+, x^-, d^+, and d^- and a number of connections between them. Following Pavlov's (1927) account for conditioned inhibition, there are two symmetrical parts in the model. The excitatory part is responsible for the initial acquisition of a response and the inhibitory part is responsible for the build-up of a connection to the inhibitory part of the network during extinction. The role of the inhibitory system is thus to cancel a response that is no longer adaptive. This implies that the state of the network is different in its naive state and when it has first acquired but later extinguished a response.

There are two types of inputs to the network. The signals marked CS_1 and CS_2 represent stimulus components and can activate the two nodes x^+ and x^- through plastic connections.

The signals through each of these connections are multiplied with their weights, w, before they are added at x.

$$x^+(t) = \Sigma_{ij}\, CS^+_i(t)\, w^+_{ij}(t). \quad (1)$$

Initially, the weights on these connections are assumed to be very small. Before any learning, a stimulus will not activate either x^+ or x^-. The other type of input controls the learning process. These signals are denoted by Rew and Pun. The output of the network, the CR, is given by the difference between the activity on the excitatory side of the network at x^+ and the activity in the inhibitory side at x^-, i. e., $CR=x^+-x^-$.

The learning dynamics during acquisition are implemented by the static connections among the four nodes in the following way. (Since the network is symmetrical, the equations for the inhibitory part are easily derived from the ones describing the excitatory side. I will only describe the equations for the positive part here.) Given that no prior learning has occurred, all plastic weights are small.

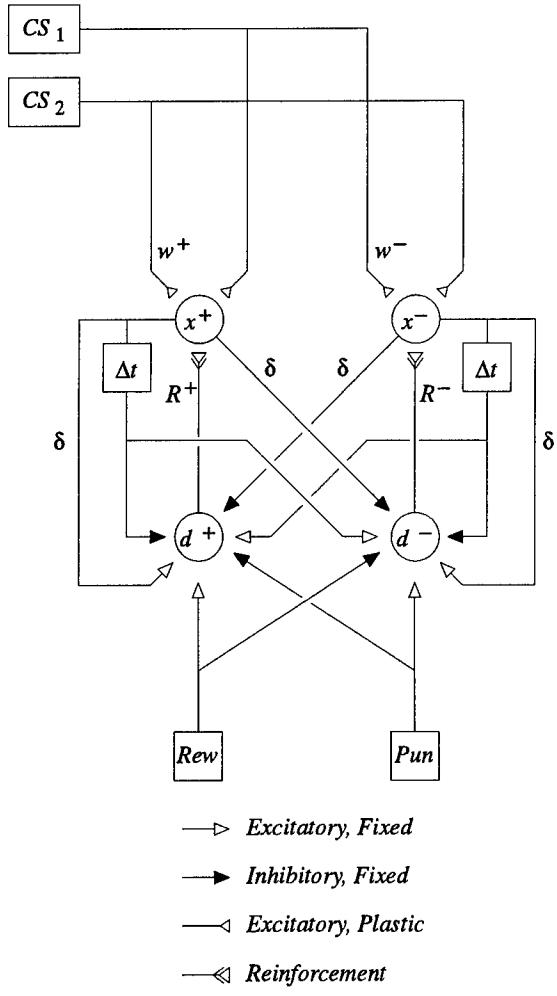


Figure 3 A neural-network model of classical conditioning. (See the text for explanation.)

The activation of, for example, CS_1 which is immediately followed by an US (or Rew) will generate activity at d^+, since,

$$d^+(t) = Rew(t) - x^+(t\text{-}1) + \delta\Delta x^+(t) + x^-(t\text{-}1) - Pun(t) - \delta\Delta x^-(t), \quad (2)$$

This equation describes a symmetrical extension of the delta-rule (Widrow and Hoff 1960/1988) which is also able to account for secondary conditioning using the $\delta\Delta x^+(t)$ and $\delta\Delta x^-(t)$ signals (see below). The activity at d^+ will generate a positive reinforcement signal.

$$R^+(t) = max(d^+(t), 0), \quad (3)$$

This signal will strengthen the connection between CS_1 and x^+ according to the following equation,

$$w^+_{ij}(t+1) = w^+_{ij}(t) + \gamma^+ R^+(t) CS_i(t\text{-}1), \quad (4)$$

where γ is the learning rate parameter.

Consistent with Rescorla-Wagner theory (Rescorla and Wagner 1972), this process will continue until the activity of x^+ becomes sufficiently strong to inhibit the reinforcement at d^+ through the time delayed connection from x^+. The signal from x^+ to d^+ can, thus, be seen as an expectation that the US will following CS_1.

The goal of the circuit is thus to let the output from x^+ agree with the reward given after the presentation of a CS. The learning process terminates when the output from x^+ has reached the same level as the Rew signal (See figure 4).

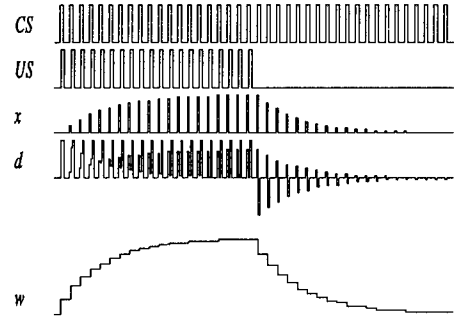


Figure 4 Computer simulation of acquisition and extinction in the neural network model described in the text. CS and US are the intensity of the stimuli and the values x and d are plotted as the difference between the activity of the two sides of the network, thus, CR = $x = x^+-x^-$, $d = d^+-d^-$. Similarly, the associative strength w is plotted as w^+-w^-.

3.2 Extinction

In an extinction experiment, an animal is first taught to respond to a conditioned stimulus, CS. Thereafter, the CS is presented on its own without the US. This will cause the CR to gradually disappear until the animal no longer responds (Mackintosh 1974).

During extinction, the role of the inhibitory side of the network becomes apparent. Since the network has previously learned that a reward follows CS_1, the presentation of CS_1 will cause the node x^+ to become active. This node will in turn activate d^- after a time delay of Δt. If this were an acquisition trial, the activity of d^- would be cancelled by the reward, but since it is now omitted, the activity of d^- will prevail and cause a negative reinforcement signal R^- to be generated. This signal will act as its positive counterpart, but this time, learning will take place between CS_1 and x^- and results in a decrease in responding.

The inhibitory learning will continue as long as the activity at d^- is larger than that at x^-. When the process comes to an end, the weight on the connection from CS_1 to x^- will have reached the same level as the weight from CS_1 to x^+, and no CR will be produced (See figure 4).

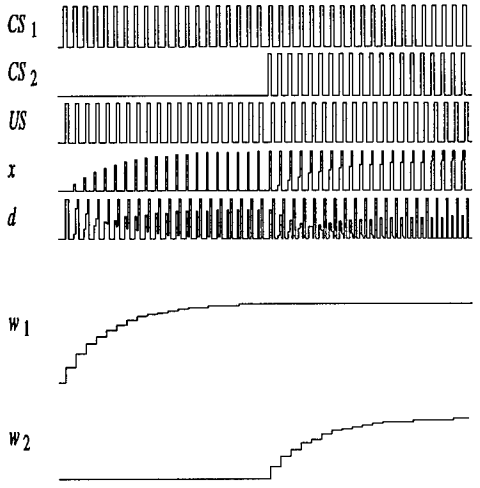


Figure 5 Secondary Conditioning. (See text for explanation.)

3.3 Secondary Conditioning

In secondary conditioning, a stimulus, say CS_1 is first paired with the US until a steady level of responding is achieved. At this point, a second stimulus CS_2 is introduced which is subsequently paired with CS_1 without the US being present. It is well known that this procedure causes CS_2 to produce the CR on its own (Mackintosh 1983). The typical explanation of this phenomenon is that CS_1 has acquired reinforcing properties during its previous pairing with the US and is now able to act as if it was a US.

This is modelled by the network in the following way (See figure 5). The activity generated at x^+ is sent to d^+ by two different pathways. The first, which was described above, is delayed by one time-step and is used to inhibit the reward. This signal causes the positive reinforcement to be shut off when the weight on the excitatory connection has reached its final value. The second pathway is excitatory with the weight δ. This signal is sent to the reinforcement node d^+ without delay and acts as secondary reward.

This signal, called Δx^+, is calculated as specified in equation (1) except that only positive *changes* in each CS is used and not its absolute level. In the simulations reported below, this means that Δx^+ is given a positive contribution only at the on-set of a CS component. In this way, the activity at x^+ is first used as secondary reward and immediately thereafter as expected reward. Since secondary reward and punishment are connected exactly as primary reward and punishment, they all share the same properties.

As discussed by Klopf (1988), it is necessary that CS_1 is followed by the US during secondary conditioning. If not, the reinforcing properties of CS_1 would gradually become extinguished. This, in turn, would cause the secondary association from CS_2 to vanish after some initially successful trials of secondary learning.

The connection weight δ, which should be less than 1, functions as a discount factor (cf. Sutton and Barto 1990). This weight makes sure that a secondary reward is never as large as the primary one. As a result, secondary reward will be worth δUS. More generally, an nth order reward will be worth $\delta^{(n-1)}US$. The result of this design is that the reinforcement module sets up a temporal gradient around the terminal event, the presentation of the US. In the next section, we will see how such a temporal gradient can be converted to a goal gradient that can guide sequential behavior.

3.4 Instrumental Learning

In an instrumental learning experiment, the animal is placed in the experimental apparatus and is allowed to behave freely. When the animal performs a certain response, it will be rewarded. This causes that specific response to become more likely when the animal finds itself in the same situation at a later time.

In Mowrer's (1960/1973) influential two-process theory of learning, instrumental conditioning was assumed to proceed in two steps. First, the positive emotional aspects of the rewarded situation would be learned by classical conditioning. In the second step, the rewarding properties of the situation were used to reinforce the appropriate behavior. Klopf, Morgan and Weaver (1993) presented a neural network architecture that can transform a model of classical conditioning into one of instrumental learning in this way. The method described is to implement the two-process idea in an artificial neural network.

Here, I will extend the reinforcement module presented above with such a network (figure 6). To admit instrumental learning, a second learning process is added between the

conditioned stimuli and responses. This second learning system is controlled by the reinforcement module described above. By shunting the connections from CS to x with the selected response, the activity in the classical conditioning network is determined by the conjunction of the currently perceived stimulus components *and* the response produced.

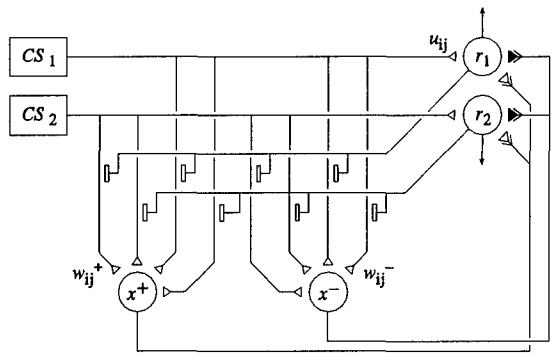


Figure 6 A neural network model of instrumental conditioning. A secondary learning system is added to the reinforcement module shown in figure 3. The signals from CS to x are shunted by the connections from the selected response. The nodes x^+ and x^- are the ones shown in figure 3.

This is modelled by replacing (1) above with,

$$w^+_{ij}(t+1) = w^+_{ij}(t)+\gamma^+ R^+(t)CS_i(t-1)r_j(t), \qquad (1')$$

where $r_j(t)$ is 1 if response j has been selected at time t, and 0 otherwise.

When the animat produces a certain behavior, the secondary learning system tries to make the connection strength between the current situation and the performed response approach that of the discounted reward of the subsequent response. In effect, instrumental learning can be seen as a case of secondary conditioning of a response to a specific stimulus configuration. Since classical conditioning is seen as an important part of instrumental conditioning, all the properties of classical conditioning carry over to this case.

The associations from stimuli to responses, u_{ij}, change according to the following equation,

$$u_{ij}(t+1) = u_{ij}(t)+\gamma[x^+(t)-x^-(t)]CS_i(t-1)r_j(t). \qquad (6)$$

Their role is to predict the discounted reward for each possible response in the current situation. These associations are used to select the response at each time step. First, the activation, ρ_j, of each response is calculated as,

$$\rho_j(t) = \Sigma_i\, CS_i(t)\, u_{ij}(t). \qquad (7)$$

The response to perform is subsequently selected according to a Bolzmann distribution generated by these activities ($\rho_j(t)$).

When the network learns a spatial behavior sequence, the associative strengths learned by the network can be seen as a goal-gradient (cf. Hull 1932). Responses closer to the goal receive larger associative strengths than responses further away. It is this decreasing gradient of secondary reward that serves to chain responses into a sequence. In this respect, the behavior of the network is very similar to the Q-learning algorithm (Watkins 1992) as well as to temporal-difference learning (Sutton and Barto 1990). The derivation of the model is described in more detail in (Balkenius 1996).

4 Simulations

A number of important questions can be asked about the model presented above. Can it exploit a multi-scale representation? Does generalization interfere with relearning? Is the network able to select different scales in different situations? To try to answer these questions, a number of simulations have been run of an animat moving in a simple grid-like environment.

The primary reason for using such an environment is that it has been much studied in the literature. It is also very easy to understand since it offers a description in terms of spatial orientation. However, this type of environment reflects the intricacies of spatial orientation only in a very superficial way, and the simulations are better interpreted in a more generic way.

To make generalization possible, each location is coded at multiple scales. A number of grids with varying resolutions generate the sensory input to the animat. The finest scale is a precise representation of the current location of the animat. This is the type of input usually employed in simulations of reinforcement learning. Grids of lower resolution generate the coarser representations (figure 7).

In the 8×8 environment, the current stimulus of the animat is coded by a binary array with 85 components, $CS_0 \ldots CS_{84}$. The finest scale uses 64 of these components to code the exact location of the animat. The next scale divides the environment into 16 regions for a slightly more blurred place representation, and so on. At all times, each scale sets exactly one of its stimulus components to 1. The remaining part of the input array is set to 0. Four components are thus active at any time.

The result of this coding is similar to the one generated by the network depicted in figure 2b. Using this coding strategy, the similarity of two locations is a function of the number of common regions at the coarser scales. Note that in this example, this is not the same as spatial proximity.

4.1 Initial Learning

A number of simulations were run with environments with 16, 64, and 256 states to compare the learning speed with and without multi-scale representations. In all simulations, the start was placed in the upper left corner of the square environment. To make generalization as useful as possible, the initial simulations used a goal in the upper right corner. The shortest path from start to goal were 4, 8, and 16 respectively. With this placement of start and goal, only the coarsest scale is necessary to solve the problem. Since the correct strategy is to move to the right at all locations along the path from the start to the goal, the finer scales are not necessary.

In all simulations, four responses where available to the animat: move-north, move-south, move-east, and move-west. During the simulation, the animat was moved back to the start location every time it entered the goal region. Each sequence of movements from the start to the goal would count as one trial.

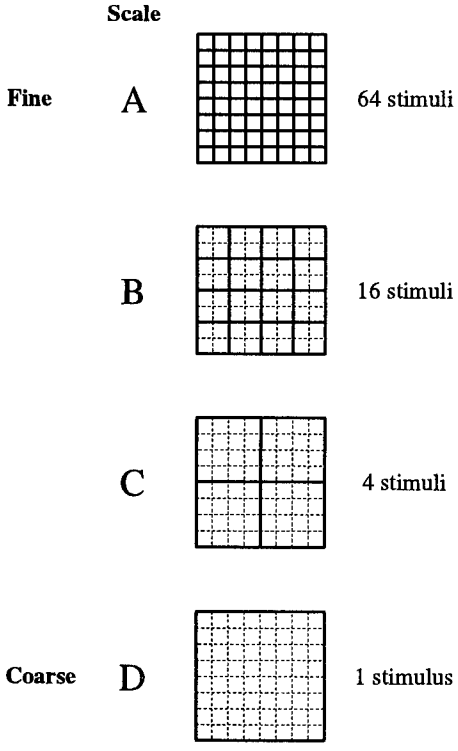


Figure 7 The coding of location at multiple scales makes generalization possible. A grid environment is simultaneously coded at four spatial resolutions. In the finest scale, 64 different stimuli are generated. At the most coarse scale, all locations in the environment fall in the same region and are coded by the same stimulus.

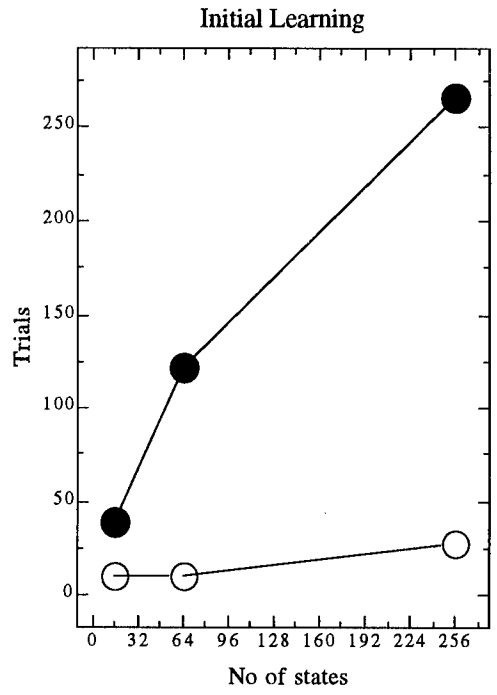


Figure 8 Learning speed with and without multi-scale representationas a function of the size of the environment (o with multi-scale representation, • without multi-scale representation). Each data point represents the average time needed to learn the environment calcualted over 10 simulations each.

Figure 8 shows the number of trials required for the animat to learn the shortest path from the start to the goal with and without generalization. The best strategy was found very fast when generalization was used. This simulation thus shows that the network is able to exploit the coarser representations when possible.

To avoid any possible advantage resulting from the larger number of components in the input array, simulations were also run in which the stimulus vector had been normalized before it was sent to the neural network. This had no effect on the learning speed which shows that the learning system satisfies the fourth requirement above in section 3.

Less satisfactory is the sensitivity to the temperature of the Bolzmann distribution. It turned out to be impossible to find a set of parameters that would work for both single-scale and multi-scale learning in the larger environment. In the simulation shown in figure 8, the temperature parameter was set to T=0.2 for the multi-scale learning and to T=0.02 when only the finest scale was used. When the scale-hierarchy is generated internally by the animat, this is not too much of the problem, but this is still a difficulty that needs to be addressed in future research.

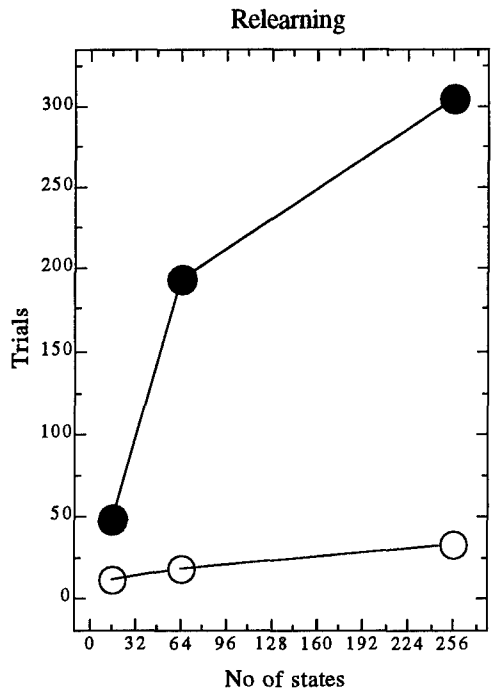


Figure 9 Speed of relearning after the goal was moved with and without multi-scale representation (o with multi-scale representation, • without multi-scale representation). Each data point represents the average time needed to relearn the environment and is the average of 10 simulations each.

4.2 Relearning

The second set of simulations investigated whether the multi-scale representation would interfere with relearning when the environment was changed. In these simulations, the initial location of the goal was first learned as above. When the animat had reached its optimal behavior, the goal was moved to the lower left corner of the environment and the network was allowed to relearn its location. As can be seen in figure 9, the ability to generalize was useful also during relearning. The network using the multi-scale representations learned the new location of the goal much faster that the one using only the fine scale. Multi-scale representations are thus useful both during learning and relearning.

4.3 Scale Discrimination

The first two simulations show that the network can utilize the coarsest scale when the best behavior strategy can be found at this scale. For generalization to be useful, however, the network must find the best scale in each situation. The final set of simulations was run to check if the network could discriminate between different scales and select the appropriate ones in a number of different environments. Each environment was learned with a variable number of active scale-representations to find how much each scale contributed to the solution of the problems.

Figure 10 shows the different environments and the changes in learning speed for the different number of active scales. Each diagram can be considered as a *scale-profile* for the corresponding environment. It shows how much each scale is used in the learning task. Note that such scale-profiles depend both on the structure of the environment *and* how that structure is represented by the animat. Recall that the sensory representation of each state is given by the arrangement described in figure 7. The environment contributes its share to the scale-profile through the location of the start and the goal together with the obstacles that constrain movement in the environment.

In the first simulation, the role of the different scales in an empty environment was tested. Although the coarsest scale (D) would in principle be sufficient to solve this problem, all scales were in fact used. As shown in figure 10, all the involved scales had an effect on the learning speed.

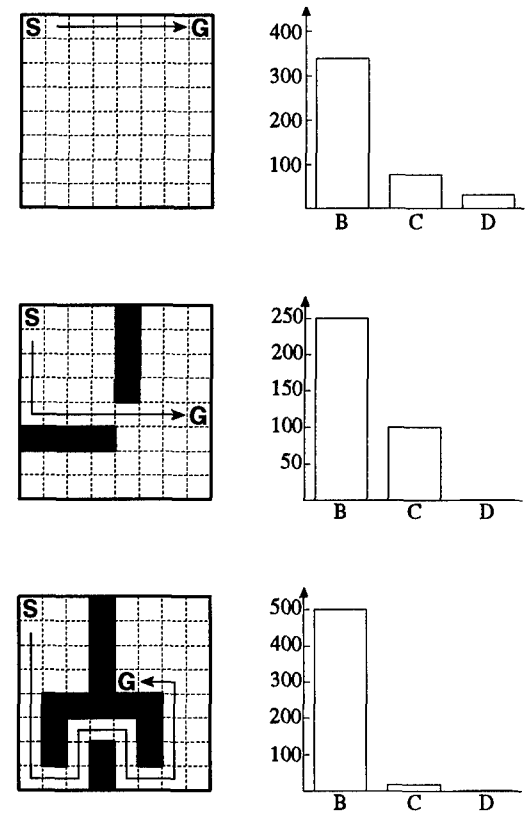


Figure 10 Scale-profiles for three different environment. The diagrams show how much faster learning became when each new scale was activated. The first bar shows how much faster the network learned when scale B was added to scale A (see figure 7). The second bar shows how much faster learning became when scale C was added to A and B, and so on.

Since the behavior performed by the animat is drawn from a probability distribution, there will always be some probability that it will deviate from the optimal path. In these cases, it can use the finer scales to direct it back to the optimal path.

In the second environment, two obstacles were introduced to see what effect they would have on the learning. As could be expected, the coarsest scale could no longer be used. However, all the other scales contributed to the solution. Finally, the model was tested in an environment with much fine structure. Here, only the two finest scales (A and B) were used to any large extent.

These profiles obviously give rather crude pictures of how the different scales are used, but they clearly show that multi-scale representations can be exploited by the learning model in instrumental learning. They do not, however, show the many complex interactions between the different scales at different locations. To investigate these mechanisms further, more advanced methods of analysis must be developed.

5 Discussion

The simulations reported above show that multi-scale representations can be very useful in instrumental learning. The important property of this type of representation is that it generates a rich set of similarities and differences (Balkenius 1994). More generally, two stimuli can be learned as similar if there exists a scale in which both are coded by the same node. The same two stimuli can also be learned as different if there exists a scale in which they are coded by different nodes.

An interesting change of perspective is to view any neural network in which a number of disjunctive nodes are connected in series as a network for multi-scale representations. When representations are viewed in this way, it becomes possible to describe a stimulus dimension, not as an objective dimension outside the animal, but instead as a collection of subjective similarities and differences. Any such collection of similarities and differences that is sufficiently dense would act as a subjective stimulus dimension, although no explicit representation of that dimension is necessary in the network. Such dimensions need not necessarily correspond in a simple way to a single observable stimulus dimension.

Another aspect of the proposed representations is that behavior can be seen as a number of overlaid behavior tendencies. At each location in the environment, every scale contributes with its own preferred behavior. The response performed is determined by the sum of these, possibly conflicting, behavior tendencies.

In the simulations above, both the environment and the responses where discrete. In the continuous case, this aspect of the model becomes even more attractive. In this approach, the goal of the learning system is to coordinate the continuous outputs of behavior modules instead of generating fixed responses, and the output from the learning system functions as parameters for an arbitration process using *additive composition* (Balkenius 1995).

This suggests that the ideas presented above could be generalized to a learning method in which the animat would first learn behavior at the coarsest scale. In situations where this would not produce the desired behavior, the finer scales could be superimposed to adjust the behavior strategy further. For each finer scale that is included, the residual error would decrease until performance reaches the desired level.

6 Conclusion

I have shown that instrumental learning can merit a lot from the introduction of scale-hierarchies in the representation of stimuli. When stimuli are represented in a way that supports a rich set of similarities and differences, generalization and discrimination become possible between a large set of stimuli.

Computer simulations have shown that this allows a neural network to learn the appropriate behavior in a grid environment much faster that if all states are coded as completely different. When similarity is represented in a scale-hierarchy, learned behavior is automatically generalized also to states that have not yet been visited. When the generalization is correct, learning will proceed much faster. If it is not, the fine scales will be used to correct the over–generalization.

In none of the performed simulations did this over-generalization make learning slower than without generalization. At worst, the generalizing system appears to do as well as the same learning system without generalization.

Acknowledgements

The support from the Swedish National Board for Industrial and Technical Development (NUTEK) is gratefully acknowledged. Thanks to Peter Gärdenfors, Lars Kopp, Janne Morén and the rest of the Cognitive Science group at Lund University for helpful discussions.

References

Balkenius, C., (1994), "Some properties of neural representations". In L. F. Niklasson and M. B. Bodén (eds.) *Connectionism in a broad perspective*, 79-88, New York: Ellis Horwood.

Balkenius, C., (1995), "Natural intelligence in artifical creatures", *Lund University Cognitive Studies 37.*

Balkenius, C., (1996), "A neural network model of classical conditioning I: The dynamics of learning", *Lund University Cognitive Studies , to appear.*

Gray, J. A., (1975), *Elements of a two-process theory of learning*, London: Academic Press.

Grossberg, S., (1987), *The adaptive brain*, Amsterdam: North-Holland.

Hull, C. L., (1932), "The goal-gradient hypothesis and maze learning", *Psychological Review*, **39**, 25-43.

Klopf, A. H., (1988), "A neuronal model of classical conditioning", *Psychobiology*, **16**, 85-125.

Klopf, A. H., Morgan, J. S. & Weaver, S. E., (1993), "A hierarchical network of control systems that learn: modelling nervous system function during classical and instrumental conditioning", *Adaptive Behavior*, **1**, 263-319.

Mackintosh, N. J., (1974), *The psychology of animal learning*, New York: Academic Press.

Mackintosh, N. J., (1983), *Conditioning and associative learning*, Oxford: Oxford University Press.

Martin, J. H., (1991), "Coding and processing of sensory information". In E. R. Kandel, J. H. Schwartz and T. M. Jessel (eds.) *Principles of neural science*, 329-340, New York: Elsevier.

Mowrer, O. H., (1960/1973), *Learning theory and behavior*, New York: Wiley.

Pavlov, I. P., (1927), *Conditioned reflexes*, Oxford: Oxford University Press.

Rescorla, R. A. & Wagner, A. R., (1972), "A theory of Pavlovian conditioning: variations in the effectiveness of reinforcement and nonreinforcement". In A. H. Black and W. F. Prokasy (eds.) *Classical conditioning II: current research and theory*, 64-99, New York: Appleton-Century-Crofts.

Sutton, R. S. & Barto, A. G., (1990), "Time-derivative models of Pavlovian reinforcement". In M. Gabriel and J. Moore (eds.) *Learning and computational neuroscience: foundations of adaptive networks*, 497-538, Cambridge, MA: MIT Press.

Thorpe, S., (1995), "Localized versus distributed representations". In M. A. Arbib (ed.) *The handbook of brain theory and neural networks*, Cambridge, MA: MIT Press.

Watkins, C. J. C. H., (1992), "Q-learning", *Machine Learning*, **8**, 279-292.

Widrow, B. & Hoff, M. E., (1960/1988), "Adaptive switching circuits". In J. A. Anderson and E. Rosenfeld (eds.) *Neurocomputing: foundations of research*, 123-134, Cambridge, MA: Mit Press.

Witkin, A., (1983), "Scale-space filtering". *8th International Joint Conference on Artificial Intelligance*, 1019-1022.

Learning to Use Selective Attention and Short-Term Memory in Sequential Tasks

Andrew Kachites McCallum
Department of Computer Science
University of Rochester
Rochester, NY 14627-0226
(716) 275-2527 FAX: (716) 461-2018
mccallum@cs.rochester.edu

Abstract

This paper presents *U-Tree*, a reinforcement learning algorithm that uses selective attention and short-term memory to simultaneously address the intertwined problems of large perceptual state spaces and hidden state. By combining the advantages of work in instance-based (or "memory-based") learning and work with robust statistical tests for separating noise from task structure, the method learns quickly, creates only task-relevant state distinctions, and handles noise well.

U-Tree uses a tree-structured representation, and is related to work on Prediction Suffix Trees [Ron *et al.*, 1994], Parti-game [Moore, 1993], G-algorithm [Chapman and Kaelbling, 1991], and Variable Resolution Dynamic Programming [Moore, 1991]. It builds on Utile Suffix Memory [McCallum, 1995c], which only used short-term memory, not selective perception.

The algorithm is demonstrated solving a highway driving task in which the agent weaves around slower and faster traffic. The agent uses active perception with simulated eye movements. The environment has hidden state, time pressure, stochasticity, over 21,000 world states and over 2,500 percepts. From this environment and sensory system, the agent uses a utile distinction test to build a tree that represents depth-three memory where necessary, and has just 143 internal states—far fewer than the 2500^3 states that would have resulted from a fixed-sized history-window approach.

1 Introduction

This paper addresses the problem of designing agents that learn to solve complex tasks by interacting with their environment through sensors and effectors. It focuses on the following question: How can a reinforcement learning agent successfully learn to interact with a complex environment when the agent's perception of that environment is restricted by physical and computational limits?

Learning agents often struggle with two opposite, yet intertwined, problems regarding their internal state space. First, the agent's state space may have "too many distinctions"—meaning that an abundance of perceptual data has resulted in a state space so large that it overwhelms the agent's limited resources for computation, storage and learning experience. This problem can often be solved if the agent uses *selective perception* to prune away irrelevant distinctions, and focus its attention on only necessary features. Second, even though there are "too many distinctions," the agent's state space may simultaneously contain "too few distinctions"—meaning that perceptual limitations, such as field of view, acuity and occlusions, have temporarily hidden crucial features of the environment from the agent. This problem, called *hidden state*, can often be solved by using memory of features from previous views to augment the agent's perceptual inputs.

In many standard approaches to learning agents, the state representation of the agent is imposed and fixed by the agent's designer—determined by whatever sensors the designer gives the agent. The agent's task is then to learn a mapping from these states to actions. However, when an agent is faced with the difficulties mentioned above, the agent cannot rely on sensory data alone to define its state representation. This paper takes the position that agents with sensory challenges should *learn* their internal state representation—by pruning the sensory space when there is too much sensory data, and by augmenting the sensory space when there is not enough. This is difficult; the agent must not only learn a mapping from states to actions, (which is often difficult enough as it is), but also learn the underlying structure of the state space itself.

1.1 Selective Perception and Hidden State

Several research efforts have addressed hidden state and selective perception individually. My perspective on these two problems is that they should be addressed together.

An agent can often have both "too much sensory data" and "too little sensory data" at the same time. For example, consider the task of driving down a crowded highway; each visual snapshot contains more features than the agent could

(or would need to) attend to at once,; however, some crucial world state information can be hidden by limited field of view, (*e.g.*, the driver must turn to see the blind-spot).

Furthermore, the issues of hidden state and selective perception are actually much more closely intertwined than one might think. Each can be expressed in terms of the other. Selective perception, in that it ignores certain features, can be seen as "purposefully hiding state." Solving hidden state with memory can be seen as a problem of "selecting" what to remember and what to forget; (*i.e.* instead of selecting from among features at one time step, with short-term memory the agent must also select from among the many possibly significant features in past time steps.)

In both cases—(1) when selecting features from the current percept, and (2) when selecting features from past percepts—the agent should try to pick out those distinctions that are relevant to the task at hand. This will result in a state space that is smaller, which is a benefit not only for the sake of efficient storage and computation, but even more importantly, a benefit for the sake of improving generalization and reducing the amount of learning experience required.

Reinforcement learning is a particularly advantageous framework in which to study task-relevant distinctions because the agent's task is clearly and mathematically defined. Reinforcement learning is a formal framework in which an agent manipulates its environment through a series of actions, and in response to each action, receives a reward value.[1] The agent stores its knowledge about how to choose reward-maximizing actions as a function that maps agent internal states to actions. In essence, the agent's task "task" is to maximize its reward over time, or *utility*. A *utile distinction test* decides whether or not to distinguish states based on a comparison of their utility.

2 The U-Tree Algorithm

This paper presents the U-Tree algorithm, a new method for dynamically building the state representation of a reinforcement learning agent. The key feature of the algorithm is that it can simultaneously handle both "too much sensory data" and "too little sensory data"—the algorithm uses selective attention to prune an unnecessarily large perceptual state space, and it also uses short-term memory to augment a perceptual state space that is missing crucial features due to hidden state.

2.1 U-Tree Ancestors

The algorithm can be seen as the result of an effort to combine the advantages of several previous algorithms: (1) Like Parti-game [Moore, 1993] and Nearest Sequence Memory [McCallum, 1995b], the algorithm is instance-based, making efficient use of raw experience in order to learn quickly and discover multi-element feature conjunctions easily. (2) Like Utile Distinction Memory [McCallum, 1993], the algorithm uses a robust statistical technique, a *utile distinction test* in order to separate noise from task structure and keep only those short-term memories that help predict reward. (3) Like the G-algorithm [Chapman, 1989], the agent can select which individual features, or dimensions of perception, to attend to.

U-Tree is a direct descendant of Utile Suffix Memory (USM) [McCallum, 1995c], which also incorporates techniques (1) and (2) above; the new feature of U-Tree is it ability to divide a percept into components, to choose to ignore some of those components, and thus to perform selective perception.

2.2 How it Works

U-Tree uses two key structures: a time-ordered chain of raw-experience-representing "instances," and a tree in which those instances are organized.

The leaves of the tree represent the internal states of the reinforcement learning agent. That is, the agent's utility estimates (Q-values) are stored in the leaves. When the agent receives an observation, it determines its current internal state by beginning at the root of the tree, and successively falling down nested tree branches until it reaches a leaf. At each non-leaf node, it chooses to fall down the branch labeled by a feature that matches its observations—working in much the same way as an exemplar is classified by a decision tree. When it reaches a leaf, the agent examines the utility estimates found in the leaf in order to choose its next action.

The key mechanism of U-Tree is the way in which it grows this tree on-line to learn a task-relevant state space. Deep parts of the tree correspond to finely distinguished parts of state space where many details and short-term memories are significant; shallow parts of the tree correspond to only loosely distinguish parts of state space where only a few features or memories are needed in order to determine what action to choose next. Another key feature of the algorithm is that both feature selection (selective perception) and short-term memory (dealing with hidden state) are represented uniformly in the same tree structure—thus capturing the inherent similarity in addressing these two problems.

The Chain and the Tree

Like all instance-based algorithms, U-Tree records each of its raw experiences. In reinforcement learning, a raw experience is a transition consisting of action-percept-reward triple, connected to its previous and successive transitions in a time-ordered chain. The linear graph at the base of Figure 1 shows an example of such a chain of instances.

U-Tree organizes, or "clusters," these instances in such a way as to explicitly control how many of their features, and how much history, to consider significant in making state-distinctions. All the instances in a cluster are used together to calculate utility estimates for that state. The structure that controls this clustering is a modified version of a finite state machine called Prediction Suffix Tree (PST) [Ron *et al.*, 1994]. A PST can be thought of as an order-n Markov model, with varying n in different parts of state space. U-Tree modifies

[1] This paper does not provide an introduction to the basics of reinforcement learning; see instead [Kaelbling *et al.*, 1995] or [McCallum, 1995a].

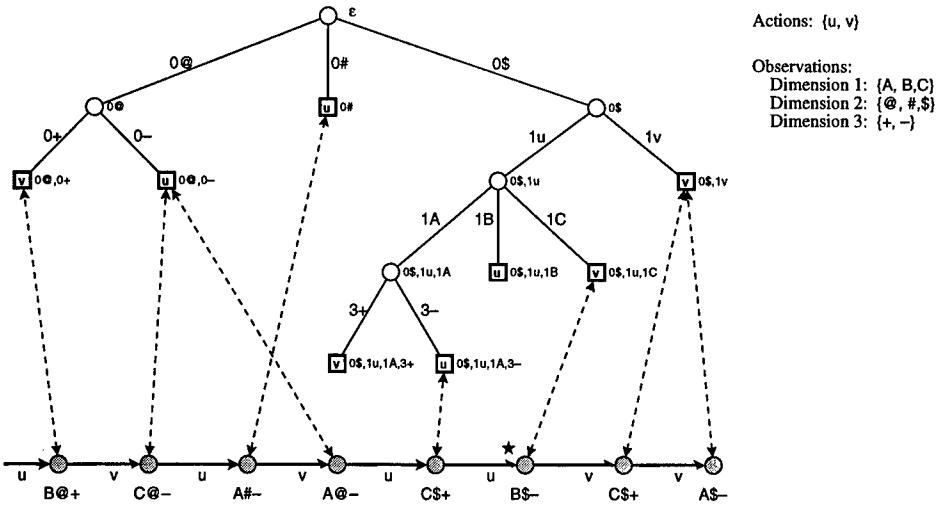


Figure 1: An example of a U-Tree instance-chain and tree, showing the relationship between instances (gray circles) and leaves (squares). The instance-chain is at the bottom; each instance represents a step in the world; every circle is labeled by the observation received at that time step, and every arrow is labeled by the action taken to make that transition. In the tree, each branch is labeled by it's history index and perceptual dimension; beside each tree node is the conjunction of features the node represents. The tree nodes drawn as squares are the agent's internal states; each contains the Q-values for each action, although the figure only shows the identity policy action. The dashed-arrows show which which agent internal state holds each instance.

this structure to allow for the consideration of individual dimensions of perception in isolation; it also adds to PST's a notion of actions and rewards. The relationship with PST's is discussed further in section 4.

A leaf of the tree acts as a bucket, holding a cluster of instances that share certain features in common. The features associated with a certain leaf are determined by the nodes on the path from that leaf to the root of the tree. Each interior node of the tree introduces a new distinction; the distinctions are based on two parameters: (1) a "perceptual dimension," indicating which individual feature (or dimension of perception) will be examined in order to determine which of the node's branches an instance should fall down, and (2) a "history index," indicating at how many time steps backward from the current time this feature will be examined. By using a non-zero history index, tree branches can distinguish between instances based on past features, and can thus represent short-term memory.

Figure 1 depicts a tree and instance chain based on a simple abstract task in which the agent can execute two actions (labeled 'u' and 'v'), and receives observations comprised of three features (labeled 'Dimension 1', '2', and '3'). For example, in a simple navigation task, actions 'u' and 'v' may correspond to turning left or right; 'Dimension 1' may correspond to a range sensor facing right, with 'A' indicating 'one foot', 'B' indicating 'three feet' and 'C' indicating 'far'; 'Dimension 2' may correspond to a range sensor facing left, and 'Dimension 3' may correspond to a bump sensor, with '+' indicating touching and '–' indicating open.

Consider the moment at which the agent has just experienced the sequence leading up to the starred instance, (observation 'B$–'). In order to find the leaf node in which the instance belongs, we begin at the root of the tree, and see that it has branches labeled '0@', '0#' and '0$'. Thus it adds a distinction based on the value of 'Dimension 2' at zero time steps backward. Since our instance has a '$' for 'Dimension 2', we fall down the right-hand branch. The node at the bottom of the right-hand branch has child branches labeled '1u' and '1v', and thus adds a distinction based on the action taken one time step ago. Since, one time step ago, the agent executed action 'u', (the transition coming into the starred instance is labeled with action 'u'), we fall down the left-hand branch. The node at the bottom of that left-hand branch has child branches labeled '1A', '1B' and '1C', and thus adds a distinction based on the value of 'Dimension 1' one time step ago. Since, one time step ago, the agent observation was 'C$+', (that is the label on the instance to the left of the starred instance), we fall down the right-hand branch, which is labeled '1C'. Here we are at a leaf, and we deposit the instance. Inside that leaf the agent will also find the policy action to be executed—in this case 'u'. The dashed line shows the association between the instance and the leaf.

Building the Tree

The agent constructs this tree on-line during training—beginning with no state distinctions (a single agent internal state, the root node), and selectively adding branches only where additional distinctions are needed. In order to calculate statistics about the value of an additional distinction, the agent builds a "fringe," or additional branches below what we normally consider the leaves of the tree. The instances in the fringe nodes are tested for statistically significant differences in expected future discounted reward. If the Kolmogorov-Smirnov test indicates that the instances came from different

distributions, then we have determined that using this distinction will help the agent predict reward (that is, in the history case, we have found a violation of the Markov property), and these fringe nodes become "official" leaves, and the fringe is extended below them, (thus, in the history case, we have turn a non-Markov state into several Markov states).

When deeper nodes are added, they are populated with instances from their parent node—the instances being properly distributed among the new children according to the additional distinction. The depth of the fringe is a configurable parameter. By using a fringe-depth greater than one, U-Tree can successfully discover useful epistatic combinations of features, in the form of multi-term conjunctions.

3 Details of the Algorithm

This section describes the steps of the U-Tree algorithm in more detail. Still further description of U-Tree can be found in [McCallum, 1995a].

3.1 Notation

The interaction between the agent and its environment is described by actions, rewards, and observations. There is a finite set of possible actions, $\mathcal{A} = \{a_1, a_2, ..., a_{|\mathcal{A}|}\}$, scalar range of possible rewards, $\mathcal{R} = [x, y], x, y \in \Re$, a finite set of possible observations, $\mathcal{O} = \{o_1, o_2, ..., o_{|\mathcal{O}|}\}$. At each time step, t, the algorithm outputs an action, $a_t \in \mathcal{A}$, then as a result receives input consisting of an observation, $o_{t+1} \in \mathcal{O}$, and a reward, $r_{t+1} \in \mathcal{R}$. We will use subscript i to indicate an arbitrary time.

The set of observations is comprised of the set of all possible values of a perceptual vector. The vector has a finite number, m, of dimensions, or "perceptual features," $\mathcal{D} = \{D_1, D_2, ..., D_m\}$. We use subscript d to indicate an arbitrary dimension index. Each feature is an element of a finite set of possible values, $D_d = \{D_{d,1}, D_{d,2}, ..., D_{d,|D_d|}\}$. The value of dimension D_d of the observation at time t is written: $o[d]_t$. Thus, we write

$$o_t = \langle o[1]_t, o[2]_t, ..., o[m]_t \rangle . \quad (1)$$

The experience associated with time t is captured as a transition "instance" in four dimensional space, written T_t. The instance is a tuple consisting of all the available information associated with the transition to time t: the previous instance in the chain, the action, a_{t-1}, the resulting observation, o_t, and the resulting reward, r_t.

$$T_t = \langle T_{t-1}, a_{t-1}, o_t, r_t \rangle \quad (2)$$

We will write T_{i-1} to indicate T_i's predecessor in the instance chain, T_{i+1} to indicate T_i's successor.

U-Tree also clusters the instances in the nodes of a tree. Tree nodes are labeled such that each node adds a distinction based on the combination of a "history index," indicating a certain number of steps backward in time, written h; and a dimension of perception or action, written d. We also include the set of actions among the the perceptual feature sets so as to allow the agent to add a distinction based on a previous action, (as in Utile Suffix Memory).

The history index of node n is written n_h; the dimension of node n is written n_d. A node, n, has as many children as $|D_{n_d}|$. No ancestor of a node, n, has both the same history index and dimension label as n. Each node of the tree can thus be uniquely identified by the set of labels on the path from the node to the root. This set, s, is called the node's *conjunction*. We will use s interchangeably to indicate the conjunction as well as the tree node that the conjunction specifies.

An instance, T, is deposited in the leaf node whose conjunction, s, is satisfied by the actions and observations of the transition instances that precede T in time. The set of instances associated with the leaf labeled s is written $\mathcal{T}(s)$. The tree leaf to which instance T belongs is written $L(T)$.

The agent maintains learned estimates of expected future discounted reward for each state-action pair. The estimate, or Q-value, for choosing action a from leaf with conjunction s is written $Q(s, a)$. Note that, although deeper layers of the tree may correspond to earlier time steps, all Q-values indicate the expected values for the next step in the future.

3.2 The Algorithm

Here are the steps of the U-Tree algorithm.

1. The agent begins with a tree that represents no distinctions. This tree has only one node—a root node.

   For this root node s, $\mathcal{T}(s)$ is empty. The time-ordered chain of instances is also empty.

2. The agent makes a step in the environment. It records the transition as an instance, and puts the instance on the end of the chain of instances,[2] *i.e.*, it creates the transition instance, T_t:

   $$T_t =< T_{t-1}, a_{t-1}, o_t, r_t > \quad (3)$$

   The agent also associates the new instance with the leaf node that whose conjunction is satisfied by this instance. We start at the root of the tree, and fall down the branch corresponding to the feature value exhibited by this instance. We continue down the tree until a leaf is reached. For this leaf, s:

   $$\mathcal{T}(s) \leftarrow \mathcal{T}(s) \cup \{T_t\} \quad (4)$$

3. For each step in the world, the agent does one step of value iteration [Bellman, 1957], with the leaves of the tree as states.[3]

[2] If one is concerned about the size of the instance chain, we can limit its growth by simply discarding the oldest instance before adding each new instance, once some reasonably sized limit has been reached. Additionally this can help the agent deal with a changing environment.

[3] If computational limitations or the number of states makes full value iteration too expensive, we can instead do Prioritized Sweeping [Moore and Atkeson, 1993], Q-DYNA [Peng and Williams, 1992] or even Q-learning [Watkins, 1989]. However, the small number of agent internal states created by U-Tree's "utile distinction" nature often makes value iteration feasible where it would not have previously been possible with typically exponential fixed-granularity state space divisions.

Value iteration consists of performing one step of dynamic programming on the Q-values:

$$Q(s,a) \leftarrow R(s,a) + \gamma \Pr(s'|s,a)U(s') \quad (5)$$

where $R(s,a)$ is the estimated immediate reward for executing action a from state s, $\Pr(s'|s,a)$ is the estimated probability that the agent arrives in state s' given that it executed action a from state s, and $U(s')$ is the utility of state s', calculated as $U(s') = \max_{a \in A} Q(s',a)$.

Both $R(s,a)$ and $\Pr(s'|s,a)$ can be calculated directly from the recorded instances. Let $\mathcal{T}(s,a)$ be the set of all transition instances in the node s that also record having executed action a. Remember that r_i is recorded as an element of T_i.

$$R(s,a) = \frac{\sum_{T_i \in \mathcal{T}(s,a)} r_i}{|\mathcal{T}(s,a)|} \quad (6)$$

$$\Pr(s'|s,a) = \frac{|\forall T_i \in \mathcal{T}(s,a) \ s.t. \ L(T_{i+1}) = s'|}{|\mathcal{T}(s,a)|} \quad (7)$$

The efficiency of the value-iteration calculation can be improved by incrementally updating these values whenever a new instance is added to the relevant tree node.

4. After every k steps, the agent tests whether newly added information or dynamic programming has changed the transition utility statistics enough to warrant adding any new distinctions to the agent's internal state space. The agent expands the fringe, and uses the Kolmogorov-Smirnov[4] test to compare the distributions of future discounted reward associated with the same action from different nodes. If the test indicates that two distributions have a statistically significant difference, this implies that promoting the relevant fringe nodes into non-fringe leaf nodes will help the agent predict reward.

5. The agent chooses its next action based on the Q-values in the leaf corresponding to its recent history of actions and observations. That is, it chooses a_{t+1} such that

$$a_{t+1} = \text{argmax}_{a \in A} Q(L(T_t), a) \quad (8)$$

Alternatively, with probability e, the agent explores by choosing a random action instead.

Increment t and return to step 2.

The main cost of the algorithm is incurred at the fringe. The most straightforward technique for expanding the fringe is to expand it by all possible permutations to a fixed depth, z, using a maximum history index of h. This however, results in

$$(h(|\mathcal{D}| + 1))^z \quad (9)$$

possible expansions for each leaf, unless some pruning occurs.

The number of expansions can be reduced considerably through a number of techniques. Here are some of the techniques used by the current implementation:

- Many leaves may contain zero or few instances. There is no need to expand the fringe below those leaves.
- Some leaves may contain many instances, but have little deviation between those instances' utility values. There is not need to expand these fringe below those leaves.
- The order of the terms in a conjunction don't matter, yet the straightforward expansion tests different orderings of the same conjunction. We choose to test only one ordering by defining a ranking on the perceptual dimensions, followed by the action dimension, and only making expansions that conform to the ranking. Note that this strongly biases the nesting order of distinctions.

All these techniques help considerably, but even so, testing the fringe is a significant computational load. Section 6 briefly describes an idea for improving this situation.

The agent only tests the distributions of the leaf node's policy action and the fringe node's policy action.

The distribution of expected future discounted reward associated with a particular node is composed of the set of expected future discounted reward values associated with the individual instances in that node. The expected future discounted reward of instance T_i is written $Q(T_i)$, and is defined as:

$$Q(T_i) = r_i + \gamma U(L(T_{i+1})) \quad (10)$$

Instead of comparing the distributions of the fringe nodes against each other, (a process that would result in n^2 comparisons, where n is the number of fringe nodes expanded under the new instance's leaf), the current implementation instead compares the distributions of the fringe nodes against the distribution in the leaf node (resulting in only n comparisons).

When a split is made, the agent starts the split test again to look for further possible splits that the new distinction may bring to light.

4 Related Work

U-Tree, like Utile Suffix Memory, inherits much of its technique and desired features from Utile Distinction Memory [McCallum, 1993] and Nearest Sequence Memory [McCallum, 1995b], but many of its ideas come from the combination of other algorithms too. Ideas from four algorithms in particular inspired the workings of U-Tree; these are: Probabilistic Suffix Tree Learning [Ron *et al.*, 1994], Parti-game [Moore, 1993], G-algorithm [Chapman and Kaelbling, 1991] and Variable Resolution Dynamic Programming [Moore, 1991]. All four of the algorithms use trees to represent distinctions and grow the trees in order to learn finer distinctions.

Probabilistic Suffix Tree. U-Tree has more in common with Probabilistic Suffix Tree Learning (PSTL) than any other algorithm. From it U-Tree borrows directly the use of a tree to represent variable amounts of memory, and the use of a fringe to test for statistically significant predictions. To this common base, U-Tree adds several features that are specific

[4] The Kolmogorov-Smirnov test answers the question, "are two distributions significantly different," as does the Chi-Square test, except that the Kolmogorov-Smirnov test works on unbinned data, a feature that is necessary since the agent is comparing distributions of real-valued numbers—expected future discounted rewards.

to reinforcement learning. We add a notion of actions, where some branches of the tree add distinctions based on previous actions. We add reward to the model, and set up the statistical test to add distinctions based on future discounted reward, not observations. We also add dynamic programming in order to implement value iteration. Also note that, unlike PSTL, the training sequences are not given to U-Tree ahead of time—U-Tree must generate the training sequences on-line using its current suffix tree.

Parti-game. The key feature U-Tree inherits from Parti-game is its instance-based foundation. Both algorithms take advantage of memorized instances in order to speed learning. Unlike PSTL, both U-Tree and Parti-game must generate their instances on-line. Parti-game differs from U-Tree in several ways. In Parti-game the agent has available a greedy controller that moves the agent towards the global goal state; new distinctions are added when the greedy controller fails. In U-Tree, the agent has no greedy controller and works with the traditional local rewards from the environment; new distinctions are added when statistics indicate that doing so will help predict return. U-Tree handles noisy rewards and actions well; Parti-game requires deterministic environments.

G-algorithm. From the G-algorithm, U-Tree inherits its ability to select individual dimensions of perception, and its use of a robust statistical test on reward values. Because of this robust test, unlike Parti-game, both U-Tree and the G-algorithm can handle noise. Unlike U-Tree and Parti-game, the G-algorithm is not instance-based. This results in a significant inefficiency whenever a distinction is added: each time the G-algorithm splits a state, the G-algorithm is forced to reset the child state's Q-values to zero. Because the G-algorithm has no raw record of the previous experience, it cannot know how to distribute the agglomerated experience from the parent state into the new children. Each time a state is split, the G-algorithm throws away all accumulated experience for that region, and must effectively re-learn that region of state space from scratch.

Variable Resolution Dynamic Programming. Unlike PSTL, Parti-game or the G-algorithm, Variable Resolution Dynamic Programming (VRDP) and U-Tree use dynamic programming directly on a learned model of the relevant parts of the environment. However, a disadvantage of VRDP is that is it partitions the state space into high resolution everywhere the agent visits, potentially making many irrelevant distinctions. U-Tree, on the other hand, creates only utile distinctions.

U-Tree is also related to Drescher's Schema Mechanism [Drescher, 1991], in that both build complex representations from simpler primitives using covariance statistics. However, U-Tree has a firm foundation in the mathematical framework of reinforcement learning, and U-Tree builds task-relevant representations.

5 Experimental Results

U-Tree performs well in a difficult highway driving task, with much utile non-Markov hidden state, and a large perception space. The agent's task is to weave in and out of one-way traffic, using actions and perceptions based on visual routines. The agent shares the road with both faster and slower trucks. The agent has control of its steering, but no control of its speed. I call the task "New York Driving." The environment has over 21,000 world states and over 2,500 sensor states; and there is also much utile non-Markov hidden state. Performance on this task strongly benefits from the combination of both selective perception and short-term memory.

This task involves interacting with a changing world and responding to severe time pressure. Many other tasks used as reinforcement learning test-beds, such as box pushing, block stacking, mazes, etc, do not have this difficulty.

5.1 The Environment

The environment consists of four lanes of uni-directional traffic. The traffic includes the agent's car and many "obstacle" trucks. The agent's car makes forward progress at a constant rate of 16 meters per second. Some trucks travel more slowly, moving at 12 meters per second; we call these "slow trucks." Other trucks travel more quickly than the agent, moving at 20 meters per second; we call these "fast trucks." The agent car has the capability to change lanes; trucks do not change lanes.

When the agent car runs into the back of a slow truck in its lane, the two vehicles do a "NYC squeeze," scraping each other's sides, but still managing to squeeze by each other within that single lane. When a fast truck reaches the back of the agent's car, the fast truck slows down to match the speed of the agent's car, and also begins beeping its horn, continuing to beep until the agent gets out of the way by shifting into another lane; at that point the fast truck continues at 20 meters per second.

Time is discrete at the resolution of one-half second per step. At each step all vehicles instantaneously change position according to their speed. During a time step in which the agent car changes lanes, it both shifts lanes and moves forward its normal distance.

New slow trucks appear in randomly selected lanes on the agent's forward horizon; new fast trucks appear in randomly selected lanes on the agent's rear horizon. At each half-second a new truck appears with probability 0.5. The new truck is chosen to be fast or slow with equal probability.

Trucks have a color, randomly chosen from red, blue yellow, white, tan, and gray. The road and shoulder have a color randomly chosen from yellow, white, tan and gray.

The agent's visual horizon is 66 meters meters in front and behind of the agent's view-point in the car. The length of road around the agent car can be seen as seventeen discrete four-meter-long sections—eight in front of the agent's car and eight behind the agent's car, and one occupied by the agent's car.

The reward function delivers one of three possible rewards at each time step. Whenever the agent scrapes by a slower truck, it receives a negative reward of -10.0; whenever a fast truck is honking its horn at the agent, the agent receives a

negative reward of -1.0; otherwise, the agent receives positive reward of 0.1 for making clear forward progress.

Action	Description
gaze-forward-left	Look at closest truck in lane to the left.
gaze-forward-center	Look at closest truck in agent's lane.
gaze-forward-right	Look at closest truck in lane to the right.
gaze-backward	Look backwards in current gaze lane.
shift-to-gaze-lane	Steer car into lane where looking.

Table 1: The actions of the agent driver

The agent navigates using five actions, which are based on visual routines and deictic actions [Ullman, 1984; Agre and Chapman, 1987; Ballard *et al.*, 1996]. These actions are: gaze-forward-left, gaze-forward-center, gaze-forward-right, gaze-backward, and shift-to-gaze-lane. Table 1 also lists the actions. The gaze-forward visual routines begin by positioning the agent's gaze immediately in front of the agent in the left, center or right lane (relative the agent's lane), then the routines trace the gaze forward along that lane until either an truck or the horizon is reached. The gaze-backward action performs the same lane-tracing behavior, except it begins immediately behind the agent and traces backward; the lane in which the agent traces backwards is the same lane (left, center or right) in which the agent was looking previously. (Thus, for example, there is no way to shift gaze from forward right to backwards left with a single action. The agent must execute gaze-forward-left, then gaze-backward.) The shift-to-gaze-lane action is a deictic action that causes the agent's car to shift into the lane at which the agent is currently looking. This action works whether the agent is looking forward or backward. As the last step of executing the shift-to-gaze-lane action, the agent's gaze is placed at the forward center of its new lane, as if it had executed an implicit gaze-forward-center.

Dimension	size	values
Hear horn	2	yes, no
Gaze object	3	truck, shoulder, road
Gaze side	3	left, center, right
Gaze direction	2	forward, backward
Gaze speed	2	looming, receding
Gaze distance	3	far, near, nose
Gaze refined distance	2	far-half, near-half
Gaze color	6	red, blue, yellow, white, gray, tan

Table 2: Sensory system of the agent driver

The agent's sensory system delivers seven dimensions of sensor information; these are: hear-horn, gaze-object, gaze-side, gaze-direction, gaze-speed, gaze-distance, gaze-refined-distance, and gaze-color. These dimensions of perception are also shown in table 2. Hear-horn is one of two values, indicating whether or not a fast truck is on the agent's tail, beeping its horn. Gaze-object indicates whether the agent's gaze is pointed at a truck, the road or the shoulder. The only situations in which the agent's gaze will fall on the road is when it looks into a lane with no trucks in the specified direction. Gaze-side indicates whether the agent's gaze is left, center or right of the agent car position. Gaze-direction indicates forward or backward. Gaze-speed indicates whether the object the agent is looking at is looming or receding. When looking forward, the road and shoulder are always looming; slow trucks also loom; fast trucks recede. Gaze distance gives a rough-grained indication of the distance to the gaze point, indicating far, near or "nose," which is closer than near. "Nose" covers distances from 0 to 8 meters, "near" distances from 8 meters to 12 meters, and "far" all the further distances. Gaze-refined-distance provides the agent with a finer grain distance measure, by dividing each of the gaze-distance regions into two. Because it is in a separate "dimension," the agent can choose to have finer grain resolution in some situations, but only rough-grained resolution in others. Gaze-color is one of six values—either red, blue, yellow, white, gray or tan if looking at a truck, and yellow, white, gray or tan if looking at the road or shoulder.

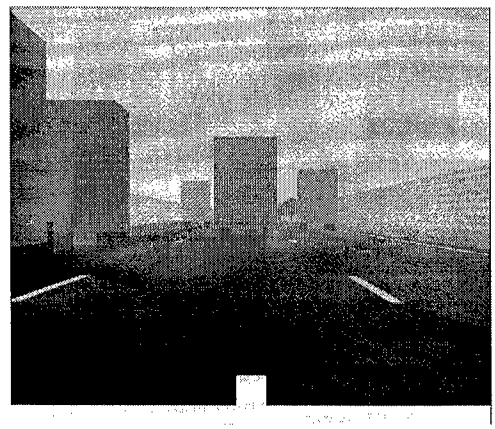


Figure 2: The perception from the driver's point of view, showing the driver's hood, the highway, and trucks.

What is the size of the sensory state space? With two possible horn states, three possible gaze objects, two possible gaze speeds, three possible sides, two directions, three distances, two refined distances and six colors, there are 2,592 possible sensor states. Note, however, that some of these sensor states never occur in this environment: whenever looking at the shoulder or the road: (1) the gaze distance is always at the horizon, (2) the speed is always exactly correlated with the gaze direction and (3) the gaze color is one of gray, white or yellow. Thus, factoring out the distance dimensions, the speed and the color for the road and shoulder gaze objects, there are actually 648 sensor states experienced by the agent. Although, note that, in general, the agent has no way of knowing this ahead of time; traditional Q-learning would still have created a Q-table of size 2,592.

```
   0 1 2 3 4 5
16 ;         ;
15 |   R<    |  PrevAction       :  Goto gaze-lane
14 '         '  Perception:
13 ;     G   ;   [0] Gaze color        :  Red
12 |         |   [1] Gaze refined dist :  Far-half
11 '         '   [2] Gaze distance     :  Far
10 ;       W ;   [3] Gaze speed        :  Looming
 9 |         |   [4] Gaze direction    :  Forward
 8 '   O     '   [5] Gaze side         :  Center
 7 ;     R   ;   [6] Gaze object       :  Truck
 6 |       G |   [7] Hear horn         :  No
 5 ' w       '
 4 ;         ;
 3 |         |
 2 ' b   B   '
 1 ;         ;
 0 |         |
```

Figure 3: An ASCII representation of a bird's eye view of the driver's environment. The 'O' indicates the agent driver's car; the other capital letters indicate slower trucks; the lowercase letters indicate faster trucks. The '<' character points at the driver's gaze point. The shoulder is drawn with '|' and ';' characters. From this position, the U-Tree driver next looked right, shifted right, then looked right and shifted right again—squeezing between the gray ('G') and white ('W') slower trucks.

What is the size of the world state space? There are four lanes, and seventeen four-meter length positions for trucks, thirteen possible occupations of each four-meter length of each lane (empty, fast truck, slow truck, and six possible colors for each truck), and four possible lanes for the agent car. The direction of the agent's sensors is also part of world state. The sensors can be pointing in three possible lanes relative to the agent, two possible directions (forward and back). All the other sensor dimensions are determined by the state of the exterior world in conjunction with the sensor direction. Without including the direction of the agent's sensor system, there are 3536 possible world states. Including the configuration of the agent's sensor system, there are 21,216 world states.

5.2 *The Task, Hidden State and Selective Perception*

Good performance on the task involves avoiding slower trucks and avoiding getting beeped at by faster trucks. To avoid slower trucks, a driver must gaze forward to check for looming trucks in its lane, and when it sees a looming truck, search the adjacent lanes for the most clear alternative lane. To avoid getting beeped at for long, the driver will also shift lanes to get out of the way of tail-gating faster trucks. This involves the same kind of lane search. Highly skilled drivers may even gaze backwards occasionally to shift out of the way of looming faster trucks even before they hear the horn once.

Note that there is utile non-Markov hidden state involved in the search for alternative lanes. In some situations seeing a truck in the left lane means that the driver should look right to find a more clear lane, but if the driver has already looked right, and the right lane contains a looming truck that is even closer than the looming truck in the left lane, the agent should shift left anyway, (and hope to have time to safely shift again later). This is one example of hidden state, there are many others.

Note that selective perception is also beneficial in this task because not all sensory dimensions are relevant in all situations. The most blatant case is that the distance and speed dimensions are not significant when seeing the road or the shoulder, because those dimensions always have the same value when looking at the road or shoulder. There are also more subtle examples of task irrelevant features, especially when considered in conjunction with short-term memory. For instance, when an agent looks behind to avoid getting beeped at, if it sees a receding (slower) truck, it does not matter how far away that truck is.

Trying to solve this task with an algorithm that makes perceptual distinctions, as opposed to only utile distinctions, would be a horrible mistake. This strategy would require an extremely large number of distinctions. For example, it would involve building memories for predicting that, if we looked right, looked left, and then looked right again we would probably see the same color truck. The agent would create a huge state space for making a plenitude of these predictions that are totally unnecessary for solving the task.

5.3 *Comparing Performance Against a Human-Written Policy in an Environment With Only Slower Trucks*

In order to obtain concrete experience with how difficult it is to solve this task, first I wrote a policy myself. In order to be fair, I coded a policy with much the same structure as the tree used by U-Tree. In a little over three hours I wrote a policy that I thought would do well when there were only slower trucks, but no faster trucks. The policy consists of nested if-then-else statements, each of which looks at one or more dimensions of perception, and has the ability to look backwards to a particular history index. The policy's if-then-else tree has 32 leaves. I found that thinking about all the additions necessary for handling faster trucks would be a much larger undertaking, and far too tedious, so I stopped there.

I modified the environment to create only slower trucks. The per-half-second new truck creation probability was still 0.5. I also modified the sensory motor system to remove those actions and dimensions that are only relevant to looking backwards; these are: the gaze-backwards action, and the gaze-direction and gaze-speed perception dimensions.

I tested my hand-written policy in this environment. Over the course of 5000 steps, my policy made 99 collisions.

An agent that chooses random actions made 788 collisions.

Then I trained U-Tree in the same environment. U-Tree was trained for 10,000 steps, or 83 simulated minutes of experience. The utile distinction test used Kolmogorov-Smirnov probability of $p = 0.0001$, the number of steps taken between splitting tests was $k = 1000$. For the first 2000 steps, the exploration probability was 1.0; for the second 4000 steps, it was 0.4; then 2000 steps at 0.2; and 2000 steps at 0.1. This exploration schedule is fairly arbitrary. I found that it was important to explore a lot in the beginning, and explore less during later steps, but I did not perform a wide array of experiments to determine the best exploration schedule. The values for p and k are also fairly arbitrary.

During a 5,000 step test run after learning, the policy learned by U-Tree made 67 collisions. This represents a 32 percent improvement of my hand-coded policy, and a 91 percent improvement over the random policy. The learned policy has 51 leaves.

The full policy tree created after 10,000 steps can be found in [McCallum, 1995a]. Features that are not relevant to the task, like color, for example, are never used to distinguish states.

Not only does the agent perform better than the policy I wrote, it also seems to drive much more aggressively. Sometimes it lets trucks get close, and swerves out of their way at the last half-second or two. I think the agent has actually learned to take advantage of the fact that, because of the way new trucks are randomly added to the environment, no two trucks will ever appear at the same distance forward from the agent. Thus, if the agent comes right up to the tail of one truck and then shifts lanes at the last half-second to arrive side-by-side with that truck, it can be guaranteed that there will be no other truck there.

5.4 *Learning to Drive Around Slower and Faster Trucks*

Although I have no human-written policy to compare against, I re-added the faster trucks, and re-added the perception dimensions I had removed, then trained U-Tree on this task.

In this environment, over the course of 5000 steps, an agent that chooses random actions made 1260 collisions and spent 775 half-seconds getting honked at.

I trained U-Tree for 18,000 steps, or two and a half simulated hours of experience. The utile distinction test used Kolmogorov-Smirnov probability of $p = 0.0001$, the number of steps taken between splitting tests was $k = 1000$. For the first 2000 steps, the exploration probability was 1.0; for the second 2000 steps, it was 0.8; then 8000 steps at 0.4, 2000 steps at 0.2; and 2000 steps at 0.1. Again, this exploration schedule is fairly arbitrary—I did not perform a wide array of experiments to determine the best exploration schedule. The values for p and k are also fairly arbitrary.

During a 5000-step test trial, the policy learned by U-Tree made 280 collisions and got honked at for 176 half-seconds. The learned policy has 143 leaves. The represents a 77 percent improvement in collision avoidance and a 77 percent improvement in honking avoidance over the random policy.

A print out of the full policy tree containing 143 leaves is too large to be included in this document. The tree and other experimental data can be obtained by contacting the author.

The emergent behaviors exhibited by the learned policy were actually quite complex and successful. As discussed on page 8, the agent learns to move out of the way of faster trucks before they start honking. The agent does this by alternately looking forward and backward whenever it sees no trucks. When it sees a slower truck in front, or a faster truck behind, the agent gets out of the way by shifting lanes. This strategy requires memory. Some particularly interesting sequences show how the agent shifts to avoid looming trucks, but stays in its lane when it sees receding trucks.

The agent learns to successfully avoid slower trucks and move out of the way of honking faster trucks. The agent, however, does not always avoid scrapes with slower trucks. Sometimes the randomly created configuration of slower trucks results in a situation in which it is impossible to avoid a scrape; sometimes the agent's lack of good exploration results in a policy that doesn't handle certain avoidable patterns of trucks. An idea for improving exploration is briefly described in section 6.

One amusing behavior the agent learns is that whenever a crash is imminent and unavoidable, the agent chooses the gaze-backwards action! I equate this with a scared person covering their eyes. I believe that the reinforcement learning mechanism behind this action choice is that looking backwards puts the agent in a perceptual state that is not *always* associated with imminent punishment. With enough training and enough data, the agent will uncover the hidden state associated with "covering its eyes" in these situations. It will add a new branch that distinguishes between looking backwards after seeing an imminent crash and looking backwards after having come from other situations.

6 Discussion

U-Tree successfully combines many different techniques instance-based learning, statistical techniques robust to noise, dynamic programming on a learned model of the environment, variable length short-memory for overcoming hidden state, selective perception for handling excessive sensory data, and utile distinctions for building task-relevant state spaces. The algorithm seems to fit many disparate technical ideas together smoothly, and the preliminary experimental results look promising.

There are several caveats and restrictions. (1) Although U-Tree handles large state spaces better than many previous techniques, there is no escape from the "curse of dimensionality," and current techniques for expanding the fringe can get expensive for very large state spaces. (2) There is a three-way "chicken and egg" problem during learning: the building of the tree (through the utile distinction test) depends on the

current utilities, the utilities depend on the current policy, the current policy depends on the structure of the tree. My experiments have used extensive exploration to try to avoid local the local minima that this circular dependency can cause. (3) Efficient exploration in the presence hidden state is a difficult issue that is completely unaddressed in this work. (4) U-Tree's "linear suffix" style of short-term memory does not directly represent loops in the environment; remembering a feature that occurred before an arbitrarily repeatable sequence is possible, but not efficient. (5) The current version of this algorithm does not handle continuous state spaces.

There are several interesting possibilities for further development: (1) The fringe branches could test more sophisticated distinctions by using complex tests supplied by the agent designer, thus providing a straightforward means for imparting domain knowledge to the agent. (2) We could modify the tree to make utile distinctions in *continuous* perception and action spaces; branches would correspond to certain threshold values in continuous space. (3) We could handle noisy perception better by explicitly representing it using techniques from probabilistic decision trees. (4) We could perform more efficient exploration by keeping exploration statistics in the nodes of the fringe. (5) We could use information-theoretic techniques, such as MDL, to aid the search for good combinations of splits; for example [Quinlan, 1995]. (6) We could avoid the use of a fringe by first clustering instances in reward space, then searching for features that separate the clusters. For more detailed explanation and discussion of all these ideas, see [McCallum, 1995a].

Acknowledgments

This work has benefited from discussions with many colleagues, including: Dana Ballard, Andrew Moore, Leslie Kaelbling, and Jonas Karlsson. This material is based upon work supported by NSF under Grant no. IRI-8903582 and by NIH/PHS under Grant no. 1 R24 RR06853-02.

References

[Agre and Chapman, 1987] Philip E. Agre and David Chapman. Pengi: an implementation of a theory of activity. In *AAAI*, pages 268–272, 1987.

[Ballard *et al.*, 1996] D. H. Ballard, M. M. Hayhoe, P. K. Pook, and R. Rao. Deictic codes for the embodiment of cognition. *Brain and Behavioral Sciences*, 1996. [To appear – earlier version available as National Resource Laboratory for the study of Brain and Behavior TR95.1, January 1995, U. of Rochester].

[Bellman, 1957] R. E. Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, 1957.

[Chapman and Kaelbling, 1991] David Chapman and Leslie Pack Kaelbling. Learning from delayed reinforcement in a complex domain. In *Twelfth International Joint Conference on Artificial Intelligence*, 1991.

[Chapman, 1989] David Chapman. Penguins can make cake. *AI Magazine*, 10(4):45–50, 1989.

[Drescher, 1991] Gary Drescher. *Made-up Minds: A Constructivist Approach to Artificial Intelligence*. MIT Press, 1991.

[Kaelbling *et al.*, 1995] Leslie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore. An introduction to reinforcement learning. In Luc Steels, editor, *The Biology and Technology of Autonomous Systems*. Elsevier, 1995.

[McCallum, 1993] R. Andrew McCallum. Overcoming incomplete perception with utile distinction memory. In *The Proceedings of the Tenth International Machine Learning Conference*. Morgan Kaufmann Publishers, Inc., 1993.

[McCallum, 1995a] Andrew Kachites McCallum. *Reinforcement Learning with Selective Perception and Hidden State*. PhD thesis, Department of Computer Science, University of Rochester, December 1995.

[McCallum, 1995b] R. Andrew McCallum. Instance-based state identification for reinforcement learning. In *Advances of Neural Information Processing Systems (NIPS 7)*, pages 377–384, 1995.

[McCallum, 1995c] R. Andrew McCallum. Instance-based utile distinctions for reinforcement learning. In *The Proceedings of the Twelfth International Machine Learning Conference*. Morgan Kaufmann Publishers, Inc., 1995.

[Moore and Atkeson, 1993] Andrew W. Moore and Christopher G. Atkeson. Memory-based reinforcement learning: Efficient computation with prioritized sweeping. In *Advances of Neural Information Processing Systems (NIPS 5)*. Morgan Kaufmann Publishers, Inc., 1993.

[Moore, 1991] Andrew W. Moore. Variable resolution dynamic programming: Efficiently learning action maps in multivariate real-valued state-spaces. *Proceedings of the Eighth International Workshop on Machine Learning*, pages 333–337, 1991.

[Moore, 1993] Andrew W. Moore. The parti-game algorithm for variable resolution reinforcement learning in multidimensional state spaces. In *Advances of Neural Information Processing Systems (NIPS 6)*, pages 711–718. Morgan Kaufmann, 1993.

[Peng and Williams, 1992] Jing Peng and R. J. Williams. Efficient learning and planning within the Dyna framework. In *Proceedings of the Second International Conference on Simulation of Adaptive Behavior: From Animals to Animats*, 1992.

[Quinlan, 1995] J. R. Quinlan. MDL and categorical theories (continued). In *The Proceedings of the Twelfth International Machine Learning Conference*. Morgan Kaufmann Publishers, Inc., 1995.

[Ron *et al.*, 1994] Dana Ron, Yoram Singer, and Naftali Tishby. Learning probabilistic automata with variable memory length. In *Proceedings Computational Learning Theory*. ACM Press, 1994.

[Ullman, 1984] Shimon Ullman. Visual routines. *Cognition*, 18:97–159, 1984. (Also in: Visual Cognition, S. Pinker ed., 1985).

[Watkins, 1989] Chris Watkins. *Learning from delayed rewards*. PhD thesis, Cambridge University, 1989.

Explore/Exploit Strategies in Autonomy

Stewart W. Wilson
The Rowland Institute for Science
100 Edwin H. Land Blvd.
Cambridge, MA 02142 USA
(wilson@smith.rowland.org)

Abstract

Within a reinforcement learning framework, ten strategies for autonomous control of the explore/exploit decision are reviewed, with observations from initial experiments on four of them. Control based on prediction error or its rate of change appears promising. Connections are made with explore/exploit work by Holland (1975), Thrun (1992), and Schmidhuber (1995a,b).

1. Introduction

Research on animats or "autonomous adaptive agents", whether simulated or realized as robots, has made progress on several fronts. A variety of core learning techniques exist for associating input stimuli with output actions to produce reasonable performance; underlying architectures include networks, CMACs, classifier systems, and tabular methods. There is also progress in incorporating generalization, internal state, and predictive modeling, among other aspects of intelligence. However, autonomy per se has not received great attention. Broadly, for an adaptive system to be autonomous implies it can cope or survive in an incompletely known and uncertain environment, perhaps accomplishing a task, independently of external control. To do well, the system must continually decide *when* and *when not* to learn. True autonomy implies that the system makes this decision on its own, according to its experiences.

The decision to learn is fundamentally a choice between acting based on the best information currently possessed versus acting other than according to what is apparently best—i.e., most remunerative—in order perhaps to gain new information that may permit higher levels of performance later. Learning risks a short-term cost—the "opportunity cost" of not doing the apparent best—in order to achieve higher returns in the longer run. Not learning risks those potentially higher returns in order to get known benefits now. The tension between learning and performance is often described as the "explore/exploit dilemma". Holland (1975) was one of the first to discuss the dilemma in connection with adaptive systems. He summarizes: "[obtaining] more information means a performance loss, while exploitation of the observed best runs the risk of error perpetuated". The dilemma crops up as soon as a learning system is supposed to have some degree of autonomy. However, because workers needed to focus on more basic questions, the explore/exploit issue was generally deferred. Now, due to the progress noted above, autonomy can move up the agenda, and with it systematic investigation of explore/exploit.

This paper will not provide any "magic bullets" for what is probably a fact of autonomous adaptive life. Instead it will discuss the advantages and limitations of a number of approaches to explore/exploit ("E/E") in an effort to identify those that most reduce the need for prior information and external decisions—and so increase autonomy. The next section states the incremental, reinforcement learning framework of the discussion. Section 3 describes ten progressively more sophisticated E/E "strategies". Section 4 relates the present work to Holland's (1975) "bandit" discussion, to Thrun's (1992) research on exploration efficiency, and to Schmidhuber's (1995a,b) work on autonomy. Section 5 offers some rough observations from the experimental program that has been undertaken. Finally, Section 6 concludes by suggesting that strategies based on error or its rate of change have promise, but that the biggest need is for further experiments and better methods of self-estimation of performance statistics.

2. Background Assumptions

The explore/exploit dilemma will be examined within the basic framework of reinforcement learning (Sutton 1991). On each discrete time-step, we assume that the system receives a stimulus vector x (one of a large set X of such vectors) from the environment, carries out an action a (one of its own set A of available actions), and receives from the environment a reward r (which varies with x and a and may often be zero). The system's objective is two-fold. On one hand, it must learn which actions maximize some measure of the reward received over time, such as the sum or a discounted sum of the rewards. On the other hand, it must act so as to accomplish the maximization.

To learn the value of actions, we shall assume the system learns a mapping X x A → P, where the P, which are *payoff predictions*, generally combine the reward r expected on the current time-step with an estimate of future reward to be received if the associated action is taken. For non-Markov (not predictable based on immediate sensory information) environments, the system may incorporate some degree of internal state (temporary memory) which when its value is combined with the external stimulus, effectively makes the decision problem Markov. For convenience we shall assume the environment is Markov unless otherwise noted.

Part of the system's job is to learn the X x A → P mapping. The other part is to use the mapping to choose actions that maximize the reward measure. Given an input x, the system may sometimes choose the action a that maps to the highest prediction; this will be termed *exploitation* (of current information). At other times it may choose some other action—termed *exploration*—in an effort to gain new information.

Uncertainty in the current prediction for an action can arise from several sources: (1) the estimation process involves gradual (conservative) adjustments, and the prediction has simply not been sufficiently sampled; (2) there may be *no* prediction yet, i.e., the action has never been tried; (3) if, as with neural networks or classifier systems, the system can generalize, fluctuations in the prediction may occur as the system settles on its "model" of the environment; (4) inputs, actions, or rewards may be affected by noise; (5) the environment may be non-Markov, so it appears to be non-deterministic. (If the environment has either of the last two sources of uncertainty, it will be termed "stochastic".) When uncertainty is high and the payoff mapping is poorly known, the system should evidently explore more than it exploits. Later, it should exploit more than explore. Choosing just when to do each is the essence of the explore/exploit dilemma as it occurs in the framework of reinforcement learning.

3. Explore/Exploit Strategies

If a system has a well-defined way of making the explore/exploit decision at each time-step, we shall say it has an E/E *strategy*. This section lists ten distinct strategies; some are often seen in the literature, others not so often or not at all. For consistency of viewpoint, the six "global" strategies discussed first are characterized in terms of a function $f(\cdot)$ which gives the probability p_1 that the system will execute a random explore trial on the current time-step, that is, the probability that it will make a uniform random selection from its set of available actions. In contrast, $p_2 = 1\text{-}f(\cdot)$ is the probability that the system will execute an exploit trial, i.e., will execute the action with the highest predicted payoff. The explore decision is made uniformly randomly for simplicity. Biasing the choice is of interest (and essential in non-toy problems (Whitehead 1991, Thrun 1992)), but this variation will be discussed later in connection with the "local" strategies where it makes most sense.

For any specific system and environment, the selection of an E/E strategy and its parametrization are naturally dependent on constraints that might include (1) maintaining system "energy" (cumulative reward less cost of operation) above a threshold; (2) maximizing some measure of reward received; and (3) operating for a finite period, or "lifetime", T. The interaction of strategies and constraints is complex and beyond the present scope, but some attempt will be made here to indicate their relationship. Note also that strategies below can always be combined to form various hybrids.

3.1 Global strategies

By "global" is meant a strategy where the argument of $f(\cdot)$ is a quantity independent of the system, such as time, or a statistic of the system's overall behavior, such as a moving average of its rate of reward intake. In the former case, the strategies do not change with the system's experience and are in that sense non-adaptive or fixed; in the latter, they are adaptive.

1. p_1 = constant.

In this strategy, the explore probability is chosen small enough to permit adequate on-line performance, yet large enough to allow fairly rapid initial learning; a value of 0.1 is typical (see, e.g., Sutton 1996). Because the system never stops exploring, it will automatically re-learn if the environment changes (e.g., if the environment's reward structure changes, or the system wanders off into substantially new territory). The main drawback of the constant strategy is that while the system can acquire complete knowledge of the environment and thus the *competence* to perform optimally, it can never actually do so, since it is required to execute explore trials at a non-zero rate. Thus, a fixed fraction of the time, the system sacrifices the difference in return between acting optimally and acting randomly; for large lifetime T, the total loss can be great.

In the constant strategy, the system's decision of "when to learn" is simple: it does explore trials with a fixed probability. However, the specific choice of that probability is made externally, and depends on knowledge of the environment, task, and T.

2. $p_1 = f(t)$.

In this strategy, $f(t)$ is typically a descending function of time, either going to zero assymptotically or after a finite period. The idea is to have a large fraction of explore trials early, when there is much to learn and little competence, followed by a shift to exploit trials later. Usually, $f(t)$ is a relaxation or annealing-like function

such as a negative exponential; it can also be a step, where p_1 has a high value such as 1.0 for a finite period τ, with value 0.0 afterward. For the strategy to be efficient, parameters must be chosen so that enough exploration occurs to achieve competence, but not so much as to continue exploration long after it ceases to be needed. If sufficient exploration occurs, the system can reach optimal performance; otherwise, it can't. A further weakness of the *f(t)* strategy is that if the environment changes, the system cannot re-learn.

As with the constant strategy, the system's learning decision is simple, given by *f(t)*. Similarly, success depends on considerable external knowledge.

3. $p_1 = f(R)$.

This is the first of the adaptive strategies. The explore probability depends on a moving or recency weighted average, *R*, of the system's reward on each time-step, *r*. The idea is simply that if you know how well you can possibly do in an environment, then as you reach that level, you should stop exploring. If the strategy can be correctly parametrized (based on the maximum possible *R*) then, as long as the environment doesn't change, the strategy should be successful. If the environment does change, then the parametrization may be wrong, and the system will perform suboptimally.

In the *f(R)* strategy, the system makes its own decision based on its estimate of *R*. The estimate may be subject to considerable fluctuation, since reward may be sparse and the reward stream is dependent on the system's "location" in the environment. On the other hand, long estimation times for *R* may make the strategy insensitive. Further, parametrizing the strategy depends on knowledge of the maximum *R* for the given environment. For reasonably complex environments, this information may be unavailable.

4. $p_1 = f(E)$.

In this strategy, the explore probability depends on the system's average prediction error, *E*. For reinforcement learning systems, the prediction error is the difference between the system's prediction of payoff *P*, and that actually received (recall that *P* combines current external reward and an estimate of future reward). The value of *f(E)* should decline with *E*, so that the system reduces its exploration rate as the error goes down. This strategy in principle improves on the *f(R)* strategy since it is not dependent on knowing the maximum *R*. Furthermore, if the environment changes, *E* will presumably go up, resulting in desirable renewed exploration. The basic idea of the strategy is that if the system's predictions are good, exploration is not needed; if they are not good, there is something further to learn.

Strategy *f(E)* is the first to be significantly affected if the environment is *stochastic* (or non-deterministic), meaning that the next values of *x* and *r* are not a deterministic function of the current *x* and *a*. As noted earlier, the condition can arise simply because of noise, for instance in the sensors. It can also arise if the environment is noiseless but non-Markov. In any event, strategy *f(E)* is "fooled" into over-exploring if the environment is stochastic, since the stochastic fluctuations—which cannot be reduced—will cause exploration even after the system has reached maximum competence. Note that making $f(E) = 0$ for *E* less then a threshold θ is not a general solution, since if θ is too large, the system may stop exploring before maximum competence is reached.

In the *f(E)* strategy, the system's explore decision depends on its estimate of *E*, which should be subject to less fluctuation than *R*. Furthermore, apart from the stochastic problem just mentioned, parametrizing the strategy is simpler than for *f(R)* since the optimum value of *E* is known: it is zero.

5. $p_1 = f(\dot{R})$.

This strategy bases the explore probability on the rate of change of average reward *with explore trials*, $\dot{R}$. Suppose the system measures its average reward on exploit trials, then does some explore trials, then measures average reward again. If there is little difference in average reward, then there's not much point in doing more explore trials. If there is a big difference in average reward, then further exploration is in order, because there is probably more to learn (note that the difference can be either positive or negative). The $f(\dot{R})$ strategy improves on *f(R)* in that it will "track" a changing environment. Compared with *f(E)*, it should be unaffected by a stochastic environment, since reward variation due to stochastic effects alone will average zero.

The system's explore decision in the $f(\dot{R})$ strategy depends on its estimate of $\dot{R}$, which may be subject to fluctuations similar to *R*. Parametrizing the strategy is simpler than for *f(R)*, since the "target" value of $\dot{R}$, zero, is known. However, coefficients in $f(\dot{R})$ would still be expected to depend on the maximum *R*, which, as noted earlier, may be unknown.

6. $p_1 = f(\dot{E})$.

Here, the explore probability depends on the rate of change of the system's average prediction error *with explore trials*, $\dot{E}$. The strategy attempts to overcome the drawbacks of the three previous adaptive strategies. It will track a changing environment, and it will not be affected by a stochastic environment if averaging periods are adequate. The basic idea is that if further explore trials do not change the average prediction error, then there is no reason to do more explore trials. If average prediction error does change, it must be because there was more to learn, either because the present environment is insufficiently known, or has changed.

In $f(\dot{E})$, the system's decision is based on its estimate of $\dot{E}$, which should be subject to fluctuations similar to E, but less subject than estimates of reward and reward change averages. Parametrization of $f(\dot{E})$ should be similar to $f(E)$.

3.2 Local strategies

The next four strategies are termed *local*. In them, the degree of exploration is based on a function of quantities associated with the system's response to the current input. The global adaptive strategies sense a condition that is a property of the whole system-environment interaction, then set the system's general explore probability accordingly. In the local (adaptive) strategies, the system senses a condition that is a property of a *niche* in its interaction with the environment, and chooses its action accordingly. The idea is that learning may be needed in certain input situations but not in others, or, more generally, that statistics associated with a given niche should determine the degree of exploration that takes place there.

7. $p_i = f(i,\{P_j\})$

In our assumed reinforcement learning model, each time the system receives an input x it responds with a set $\{P_j\}$ of predictions of the payoff available for each action that it might take, then selects one of the actions, which we denote a_i. The set $\{P_j\}$ might be the output of a neural network (or set of networks), a classifier system, or a table, etc. Each prediction is a statistic, based on prior experience in situations (apparently) like the current one. The general idea of the $p_i = f(i,\{P_j\})$ strategy is that actions with higher associated predictions should be selected with higher probability.

A well-known example is the so-called "roulette-wheel" selection employed in some classifier systems (Goldberg 1989). For each classifier that matches the current input, the probability that its action will be selected as the system's action is just the classifier's prediction of payoff divided by the sum of the predictions of payoff of all the matching classifiers. That is, the predictions are normalized and treated as selection probabilities. Roulette-wheel selection achieves a mix between exploration and exploitation. Higher predicting actions tend to be selected over lower predicting ones, but the latter still continue to be tried.

The $f(i,\{P_j\})$ strategy is a compromise, a bit like the constant strategy (strategy 1). It can achieve fairly good performance and, if the environment changes, it will adjust its predictions and reach fair performance anew. The drawback is that even though the system can acquire full competence, it can never realize that competence in performance, since suboptimal actions continue to have a finite probability of selection.

8. $p_i = f(i,\{P_j\},\{E_j\})$

In this strategy the probability of selecting action a_i on a given time-step depends on both the prediction for that action and the estimated error in the prediction. The idea is to select the action for which the sum (or other increasing function) of the prediction and error is high. Then, as further exploration refines the predictions and reduces their error, the system will likely end up always selecting the action for which the prediction is in fact highest. Thus, as exploration occurs and establishes true values for the predictions, the system will in effect gradually switch over to full exploitation.

An $f(i,\{P_j\},\{E_j\})$ stategy was first implemented for learning based on tables by Kaelbling (1990), who termed it the "interval estimation" strategy for its relationship to a method in statistics. Goldberg (1988) proposed a similar strategy for classifier systems called "variance-sensitive bidding", in which the system samples, for each action, a gaussian probability distribution with mean P_j and variance gE_j, and chooses the action with the largest sample value. The "gain" g determines the amount of bias toward exploration.

These strategies have the ability to sense the degree of convergence to reliable predictions and to increase exploitation. Due to their sensitivity to error, they will track a changing environment. For the same reason, they will waste trials if the environment is stochastic. As a practical matter, it should be noted that the strategies are most readily implemented in classifier systems or tables, since both error and prediction statistics are required. Because a classifier is a "record" structure like a table entry, it can easily accomodate an additional "slot" for the error. Neural networks, on the other hand, have no obvious way to estimate errors apart from doubling the size of the network or adding a second one devoted to that purpose.

Besides wasting trials in a stochastic environment, a further potential drawback of the $f(i,\{P_j\},\{E_j\})$ strategy arises because it inherently samples actions with a bias toward higher predicting ones. If the system (as in neural networks and recent classifier systems) tends to form generalizations over the input space, biased sampling can result in persistent overgeneralizations, simply because potential counterexamples are not tried, or tried too little. In principle, the next strategy avoids this problem.

9. $p_1 = f(\{E_j\})$

In this strategy, as above, the E_j are the error estimates associated with the actions available for the current input x. The explore probability, however, as in the global strategies, is simply the probability of selecting one of the actions at random (versus choosing the action with highest P_j). The strategy would be implemented by basing p_1 on the sum of the E_j, or their average. The idea

is to explore to the extent that the current situation has large error estimates, but the choice of explore action is random, in order to avoid the potential sampling bias noted above.

10. $p_i = f(i,\{P_j\},\{\dot{E}_j\})$ *or* $p_1 = f(\{\dot{E}_j\})$

These strategies are similar to 8 and 9, but replace the dependence on E_j with dependence on $\dot{E}_j$ in order to deal with stochastic environments. In a classifier system, each classifier would keep an estimate not only of its prediction and the error in that prediction, but of the rate of change of the error, $\dot{E}_j$. That is, $\dot{E}_j$ is a moving or recency-weighted average of the difference between the current error and the error estimate. Purely stochastic effects would not show up in $\dot{E}_j$ (if the averaging period was sufficient). A large value of $\dot{E}_j$ would mean that trials of a_j in that context (x) yielded substantial change in the prediction error—thus more trials should be made there. A small value of $\dot{E}_j$ would suggest that the current value of P_j was as accurate as can be determined.

This completes the list of strategies. We turn now to three authors who have made important contributions to the explore/exploit discussion.

4. Related Work

4.1 Holland's bandit model

The operation of a genetic algorithm (GA) incorporates an explore/exploit tradeoff, since the probability that an individual will be selected for reproduction is an increasing function of its fitness relative to the mean fitness of the population. The GA searches regions of the problem search space near (with respect to genetic operators) the locations of the selected individuals. Thus selection of higher fitness individuals means searching their parts of the space more heavily, a kind of exploitation of the best. Conversely, to the extent lower fitness individuals are also selected, the GA is exploring other regions "just in case" they contain unexpectedly higher fitness individuals than currently estimated.

Holland (1975, Ch. 5) seeks to determine the degree of optimality of the GA's selection mechanism. To do this he investigates the "two-armed bandit" (two slot-machine) problem of statistical decision theory, and shows that the optimal (loss-minimizing) allocation of trials to the observed better (higher-paying) arm is very nearly an exponentially increasing function of the number allocated to the observed worse arm. Taking this solution as the optimal way to allocate trials between competing uncertain alternatives, Holland shows that, in a GA, selection proportional to relative fitness is like the derivative of an exponential and thus implies that the GA is searching near-optimally. (See De Jong (1975) for a good discussion of Holland's assumptions and derivation; also, Mitchell (1996) for a review of subsequent critiques).

How are Holland's results useful for autonomous learning? Only indirectly. For one thing, the optimal allocation for the bandit is based on knowledge of the "observed best" alternative, which can't be known for sure until *after* all trials are over, when it's too late to perform the allocation. How to allocate trials as one goes along is less understood, though De Jong (p. 33) does simulations suggesting that allocation with probability proportional to the product of relative payoff and the *previous* time-step's allocation probability (which parallels the way the GA works) can approach the bandit optimum for large numbers of trials. In the present context, this idea might be applied by modifying strategy 7 to make each selection probability proportional to the current prediction times the probability of that selection the last time around (implying storage of the latter).

The major problem, however, is that Holland's allocation formula contains parameters dependent on the statistics of the bandit process, including the means for the two arms, the variances, and (in an improvement of the formula (Holland, 1992)), higher moments, apparently. Although the parameters are not required in order to know that the form is exponential, they *are* required to actually carry out the allocation—even ex post facto—of the trials! In contrast, the adaptive strategies discussed in Section 3 make no assumptions about payoff statistics, but instead attempt to estimate both the means and in some cases the variances (errors). Though the strategies have as yet only scant connection to theory, they are practical for autonomous learning in a way that the bandit results are not.

4.2 Thrun on exploration efficiency

The present article overlaps in several respects with Thrun (1992), in which the emphasis is on exploration techniques that are efficient in the sense of minimizing the costs (time-steps, negative payoffs, etc.) of learning an environment to a given performance criterion. Thrun distinguishes *undirected* exploration—in which actions are selected either uniform randomly as in strategy 1, or with probabilities related to action utilities as in strategy 7—from *directed* exploration, which aims to choose actions that will maximize information gained, as in Kaelbling's (1990) interval estimation or Sutton's (1990) recency-based exploration. The discussion of these exploration techniques is very comprehensive.

Thrun's thesis, demonstrated in two (simulated) robotics experiments, is that efficient exploration benefits greatly not only from the use of directed techniques, but also from the right amount of exploitation—not so much as to waste time exploring, but not so little as to take too long to learn the task. His demonstrations involve a framework in which exploration and exploita-

tion "desires" (utility measures) compete to set the system's action. For example, action a_1 may have a high predicted payoff (exploitation utility), but a_2 may have a high exploration utility because it has never been tried. The balance between these "desires" is set by a task-dependent parameter Γ. Though focussed on efficiency of exploration, Thrun's framework can also be considered a (local) explore/exploit strategy, or set of strategies, providing a different approach to the objectives of the strategies of Section 3.2.

However, Thrun's article emphasizes exploration efficiency for learning a task—meaning acquiring the competence to accomplish it—after which the system will be switched to exploit mode. The present paper places the emphasis on autonomy, and the continual on-line issue of when and when not to learn in possibly changing or stochastic environments. This has in particular motivated consideration of strategies based on the rates of change of performance and error statistics.

4.2 Schmidhuber's autonomy proposal

Recently, Schmidhuber (1995a,b) introduced an intriguing, if sweeping, model for environment- and method-independent autonomous learning that purports to solve the explore/exploit problem more or less *en passant*. Central to the model is the concept of *reinforcement acceleration*. The system continually applies "policy modification processes" (PMPs) to its policy (the mapping from inputs and internal states to outputs and new internal states), and measures the resulting change in the rate of reinforcement intake. The aim of the PMPs is to achieve reinforcement acceleration, meaning that the rate of reinforcement intake since application of the PMP exceeds that since application of any preceding PMP. Upon completion of a current PMP, information sufficient to restore the previous policy, plus other information about the current PMP, is pushed onto a stack. Subsequently, the system tests to see if the current PMP satisfies the *reinforcement acceleration criterion* (RAC). If so, it is retained until the next PMP occurs. If RAC is *not* satisfied, the system pops the stack, restoring previous policies, until an older PMP is reached such that its rate of reward intake, *measured from its inception up to the current time*, exceeds that for the PMP below it on the stack. Thus the only PMP (records) maintained on the stack are those for which the reinforcement acceleration criterion is satisfied. Schmidhuber offers a proof sketch that this will be the case, independent of the "(possibly changing) environment".

The PMPs can be *any* policy-modification processes consistent with the system's architecture. For instance, for a Q-learning system based on a state-action table, a PMP could simply randomly mutate the table contents. All that is necessary is that the previous policy be restorable. Still, one worries that such a general framework, though capable of improvement and maybe eventual optimal performance, would improve impractically slowly, or would require an unacceptably deep stack. It sounds much like "random mutation with preservation of the best", a technique that experience has found impractical for most problems of interest. However, Schmidhuber presents experiments in which, though times of order 10^9 time-steps are required, substantial learning occurs. In addition, for a specific implementation of the concept, he introduces a low-level, general programing language that contains "self-referential" instructions. Schmidhuber argues, and claims his experiments show, that via these instructions, the PMPs (also expressed in the language) not only improve the system policy as predicted, but let the system adaptively determine its own rate of applying PMPs and for how long. Since to apply a PMP is in effect to engage in exploration (versus further exploitation of the policy that resulted from the previous PMP), Schmidhuber argues that the system adaptively controls the explore/exploit tradeoff.

It is possible that, besides its theoretical interest, Schmidhuber's all-encompassing approach will yield practical autonomous systems in which, as a distinct issue, the explore/exploit problem simply disappears! This will depend at least on how rapidly the PMP/RAC algorithm can bootstrap itself and also how accurately the system can in practice measure its rate of improvement in challenging environments. Meanwhile, there is a resemblance between the improvement criteria of Section 3 and Schmidhuber's RAC that suggests his theory can be an important guide to other approaches to the explore/exploit problem.

5. Experimental observations

Experiments have been conducted on several of the E/E strategies using XCS, a novel kind of classifier system in which the fitness of individual classifiers is based not on their payoff prediction as in traditional classifier systems, but on the *accuracy* of that prediction (Wilson 1995). XCS is appropriate for E/E experiments on the ten strategies because its classifiers keep both prediction and error estimates (and error rate-of-change statistics can easily be added). In addition, XCS inherently evolves classifiers whose conditions are maximally general subject to an accuracy criterion on the prediction estimate, thus making it possible to evolve a generalization model over the payoff "landscape". Finally, in contrast to earlier classifier systems that employed a roulette wheel action-selection strategy, XCS can employ any strategy of the sort represented in Section 3.

Two environments from Wilson (1995) were used. One was the "6-multiplexer", a Boolean function for

which the system should learn to give the correct output of the function (0 or 1) upon presentation of a random 6-bit string. The other was "Woods2", a grid-like environment in which the system, started at a random cell, should learn to reach "food" sites in the minimum number of steps. As demonstrated in Wilson (1995), XCS does indeed learn (non-autonomously) both of these tasks when given sufficient explore trials. That is, the experimenter permits XCS n explore trials, then switches it to exploit trials and finds perfect performance if n was sufficiently (and not outrageously!) large. In addition, XCS evolves classifiers whose conditions generalize over states of the environment that have the same payoff prediction for a given action.

The current explore/exploit experiments have so far investigated versions of global strategies 4 ($p_1=f(E)$) and 6 ($p_1=f(\dot{E})$), and local strategies 8 ($p_i=f(i,\{P_j\},\{E_j\})$) and 9 ($p_1=f(\{E_j\})$).

For strategy 4, p_1 was set equal to the lesser of 1.0 and the product of a gain factor times E, a moving average of the absolute value of the difference between predicted and actual payoff. For strategy 6, p_1 was equal to the lesser of 1.0 and the product of a gain factor times $\dot{E}$, a moving average of the absolute value of the rate of change—with explore trials—of E. This rate of change was calculated by a more elaborate routine which found the difference between moving averages of E before and after a "bout" (series, usually 100, with no intervening exploit trials) of explore trials. Then p_1 for this strategy was the probability of initiating another bout of explore trials, instead of continuing with exploit trials.

Strategy 8 was an implementation of variance-sensitive bidding (VSB). On each time-step, and for each action a_i, the system sampled a gaussian distribution with mean equal to the predicted payoff, P_i, and variance equal to the product of a gain factor and the error estimate, E_i, for that action. Then it executed the action for which the sample value was largest. In Strategy 9, the system on each time-step calculated a mean of the error estimates for each action, then set p_1 equal to the lesser of 1.0 and the product of a gain factor times the mean of the error estimates.

We are not able to include statistically reliable results, since the experiments are incomplete at the time of writing. But several qualitative observations can be made.

1. All four strategies "worked" in the sense that, starting with no prior knowledge of the environments, on-line performance improved from random to near optimal levels and the incidence of explore trials fell in general to low levels. However,

2. Progress was sensitive to the gain factor. Larger values caused more exploration to occur early on, when little was known, resulting in more rapid accumulation of competence but less rapid achievement of high on-line performance as error (or error rate of change) fell. In contrast, if the gain was lower, good on-line performance came sooner but optimal levels took longer to reach.

3. The gain setting had a further consequence. For large values, the system could be very sensitive to temporary *increases* in error (or error rate of change) once good on-line performance was achieved. Greater sensitivity meant that more explore trials would be undertaken to correct a given degree of error, resulting in a greater temporary fallback in on-line performance. In contrast, lower gain settings were less sensitive to such error "blips".

4. Error (or error rate of change) "blips" occurred under all four strategies. Though progress toward low error via exploration is definite, it is bumpy. Assuming that the strategies are basically sound, one would like to be sure that the system's error *measurements* are as reliable as possible.

5. There is a tension between modeling and performance reminiscent of that between exploration and exploitation. Larger gain factors—more exploration—result in greater development of generalizations within a given number of time-steps. The price for this is the on-line performance cost noted in (2) and (3).

6. Strategy 8 (VSB) implements a biased sampling of alternate actions. As suspected, this resulted in a biased generalization model in which overgeneral classifiers evolved that retained high fitness because the actions that would contradict their payoff predictions were rarely tried. There does not as yet appear to be any obvious offsetting performance superiority for VSB compared with the other strategies.

Among the strategies tested, it is perhaps most encouraging that strategy 6, which depends on the rate of change of error, is no worse than the others. This is important, because strategy 6 is the only one that is in principal unaffected by either a changing or a stochastic environment.

Broadly speaking, the explore/exploit dilemma does not go away via the strategies tested (or, surely, the others), but they may succeed in managing it "up to a constant"—the gain parameter mentioned. In a sense, the gain parameter measures the amount of *risk* the system is willing to take for potentially higher future performance, and thus summarizes the trade-off in a single number. Where does that number come from? If it is a reality in animals, it comes from evolution, which "knows about" their environments. In artificial systems it now comes from the designer, but future research may show how it can be brought within the system's own adaptive compass.

6. Conclusion

Much further research is needed on explore/exploit strategies. The issue is crucial for true autonomy—consider the Mars robot or any system in a substantially unknown environment that cannot be teleoperated. This paper suggests that strategies based on error or its rate of change can permit systems to control the tradeoff autonomously. Major desiderata now are further experiments—to build up experience with these (and other) strategies, and greater theoretical understanding of how a system can tell—for instance via better estimation techniques—how well it's doing.

References

De Jong, K. A. (1975). *An analysis of the behavior of a class of genetic adaptive systems.* Ph.D. Thesis, Department of Computer and Communication Sciences, The University of Michigan, Ann Arbor. Available from University Microfilms International, Ann Arbor, Michigan.

Goldberg, D. E. (1988). Probability matching, the magnitude of reinforcement, and classifier system bidding (Technical Report TCGA-88002). Tuscaloosa, AL: University of Alabama, Department of Engineering Mechanics. (Also *Machine Learning, 5*, 407-425.)

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning.* Reading, MA: Addison-Wesley.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems.* Ann Arbor: University of Michigan Press. Second edition 1992, Cambridge MA: The MIT Press.

Kaelbling, L. P. (1990). *Learning in Embedded Systems.* Ph.D. Dissertation, Stanford University. Teleos TR-90-04. Also Cambridge, MA: The MIT Press/Bradford Books, 1993.

Mitchell, M. (1996). *An Introduction to Genetic Algorithms.* Cambridge, MA: The MIT Press/Bradford Books.

Schmidhuber, J. (1995a). On learning how to learn learning strategies. Technical Report FKI-198-94 (revised), Fakultät für Informatik, Technische Universität München, Munich, Germany.

Schmidhuber, J. (1995b). Environment-independent reinforcement acceleration. Technical Note IDSIA-59-95, IDSIA, Lugano, Switzerland.

Sutton, R. S. (1990). Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Proceedings of the Seventh International Conference on Machine Learning,* (pp. 216-224). San Mateo, CA: Morgan Kaufmann.

Sutton, R. S. (1991). Reinforcement learning architectures for animats. In J.-A. Meyer & S. W. Wilson (eds.), *From Animals to Animats: Proceedings of the First International Conference on Simulation of Adaptive Behavior* (pp. 288-296). Cambridge, MA: The MIT Press/Bradford Books.

Sutton, R. S. (1996). Generalization in reinforcement learning: successful examples using sparse coarse coding. In *Advances in Neural Information Processing Systems 8.* Cambridge, MA: The MIT Press/Bradford Books.

Thrun, S. B. (1992). The role of exploration in learning control. In D. A. White and D A. Sofge (eds.), *Handbook of Intelligent Control: Neural, Fuzzy and Adaptive Approaches.* Florence, Kentucky: Van Nostrand Reinhold.

Whitehead, S. D. (1991). Complexity and cooperation in Q-learning. In *Proceedings of the Eighth International Workshop on Machine Learning,* (pp. 363-367). San Mateo, CA: Morgan Kaufmann.

Wilson, S. W. (1995). Classifier fitness based on accuracy. *Evolutionary Computation, 3*(2): 149-175.

Learning Control Composition in a Complex Environment

E. G. Araujo and R. A. Grupen
Laboratory for Perceptual Robotics *
Department of Computer Science
University of Massachusetts
Amherst, MA, 01003
araujo@cs.umass.edu, grupen@cs.umass.edu

Abstract

In this paper, reinforcement learning algorithms are applied to a foraging task, expressed as a control composition problem. The domain used is a simulated world in which a variety of creatures (agents) live and interact, reacting to stimuli and to each other. In such dynamic, uncertain environments, fast adaptation is important, and there is a need for new architectures that facilitate on-line learning. Recently, control composition has shown its potential in reducing the complexity of learning in many domains, permitting more complex tasks to be addressed. Results presented here restate the above claim and also show that the use of a modified version of the Q-learning algorithm performs better in the non-Markovian sequential decision task proposed. In addition, it is shown that shaping can be successfully used to improve the performance of these learning algorithms in complex environments.

1 Introduction

Fast adaptation is crucial to any agent that interacts with a dynamic, uncertain environment. In several domains such as artificial life and robotics new architectures that ease the learning process in order to produce on-line responses are required (Maes and Brooks, 1990; Sutton, 1990; Lin, 1993). Recently, the interest in learning of control compositions has increased (Mahadevan and Connell, 1990; Singh et al., 1994). Controllers ease the learning task by compressing the state space, consequently allowing the solution of more complex tasks (Grupen et al., 1995; Huber et al., 1996). Moreover, controllers can be designed to avoid unsafe situations during the learning process and to incorporate explicitly the dynamic constraints of the agent, freeing the learning process from dangerous and complex learning situations. This paradigm defines a new set of problems, as for example, how to synthesize and compose controllers.

In this paper, two reinforcement learning algorithms applied to a control composition task are compared. Reinforcement learning algorithms learn an optimal control policy - the best action in each state - for a sequential decision problem by receiving delayed reinforcement (Barto et al., 1990). The reinforcement learning framework was selected mainly because of the small amount of supervision necessary for learning, and the fact that the reinforcement can be delayed. These are very important characteristics, allowing for the solution of problems without complete knowledge of the processes involved. For example, animal conditioning can be framed as a reinforcement learning tasks (Sutton and Barto, 1990).

The task selected to demonstrate the advantages of control composition over learning from scratch is foraging in a simulated environment, where prey are guided by a single controller, and the learning agent (predator) composes all the controllers available. The control basis selected and the idea of using a control composition approach on this task were inspired by the work of Valentino Braitenberg (Braitenberg, 1984). For a review of the challenges presented by foraging tasks on the reinforcement learning framework, see (Watkins, 1989).

The control composition model is introduced in Section 2. Section 3 describes the task, the specifics of the control composition model for the task, and the two simulated environments where the task is performed. The learning methods and the results obtained are presented in Section 5. Section 6 introduces a strategy based on shaping to facilitate learning. Finally, Section 7 presents conclusions and future work.

2 Control Composition

There are several advantages to solving a task by activating controllers instead of learning sensorimotor commands from scratch. By selecting a suitable set of controllers, the state space of the learning task is dra-

*http://piglet.cs.umass.edu:4321

matically compressed, and the resulting higher performance adaptive algorithm can be applied to more complex tasks. In addition, controllers can more easily deal with the dynamics of the agents. To learn dynamics from scratch usually requires a large state space resulting in a low quality solution considering the amount of time required for learning. In addition, controllers can focus the learning on the relevant situations and avoid the danger of exploring unsafe states. Another motivation for control composition is that behaviors considered to be complex by an observer can be generated by a composition of very simple control laws (Braitenberg, 1984).

One may argue that learning from scratch will produce better control policies. The argument made here is that this could be the case, however it will in general take more time, and it will make the learning of more complex tasks impractical. The main idea is to create a control basis able to span a wide variety of behaviors by control composition, and to obtain more complex behaviors by successive composition of previously learned controllers.

The diagram in Figure 1 is a general control composition model. Its components are loosely related to biological structures: the initial set of controllers can be associated with reflexes, the mechanism for control adaptation can be the product of evolution, and the further composition of controllers can be associated to lifetime learning. Notice that the control level could be augmented over time with the controllers previously learned by control composition, allowing a continuous increase in complexity of the behaviors that the agent exhibits.

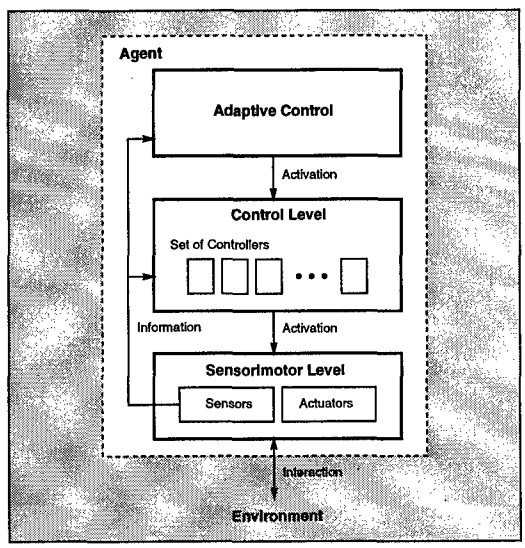


Figure 1: Control composition model of an agent.

3 Composition Task Description

The learning domain consists of a simulated world, described in Section 4, in which a variety of creatures live and interact, reacting to each other. The behaviors that the creatures display and the corresponding controllers are described in Section 3.2.

The task is foraging: by activating controllers, a learning agent (*predator*) has to search for other creatures (*prey*), approach them, and engage in a precise attack. If successful in this sequential decision task, the learning agent receives a positive reinforcement and an extra amount of energy (*food*) for each successful attack. Energy is consumed over time, and a trial ends when there is no energy left.

This problem can be described as a stochastic sequential decision task. It is non-Markovian, because of the presence of partially observable states (hidden states) in its discrete state representation. Partially observable states are caused by the limited amount of sensory information available to the agent, and also by the discretization of the state space.

The control composition model used to address the foraging task is shown in Figure 2, and its three levels are described in the following sections.

3.1 Sensorimotor Level

The sensorimotor level receives as input the torque commands to the actuators, and relays the internal and external sensor readings to the control and adaptive levels.

All creatures in the environment are inertial bodies (vehicles) with two light sensors coupled to two drive motors through some simple control logic as in Figure 3. Each creature is perceived as a light source by the others, and their light intensity sensors have a limited range of perception. In addition, the learning agent is capable of identifying the closest creature's type, relative distance at three relative depths, front, and side. Furthermore, the learning agent can perceive its own energy and internal reinforcement also in discrete levels.

3.2 Control Level

The control level receives as input a selection signal, indicating, in a mutually exclusive way, which controller to activate, and the light intensity detected by each sensor. It outputs the torque commands to each wheel, based on the control law of the activated controller.

The five different controllers used were inspired by the behaviors in (Braitenberg, 1984), and implemented as control laws as shown in Figure 3. τ_1 and τ_2 are the torques applied to each respective motor, i_1 and i_2 are the light intensity captured by each light sensor, and τ_{max}, τ_{min}, τ_{turn}, and *gain* are constants representing: the maximum torque, a small torque disturbance to avoid straight paths (allowing a better exploration of the environment), and the gain applied to transform light intensity into torque commands, respectively.

Each controller produces a different behavior: *Sleep*

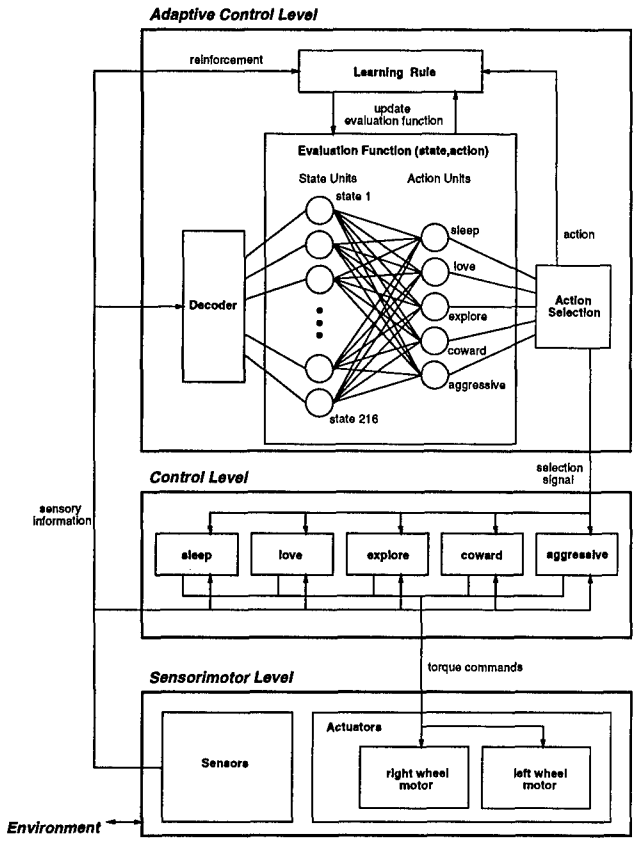


Figure 2: Task specific control composition model

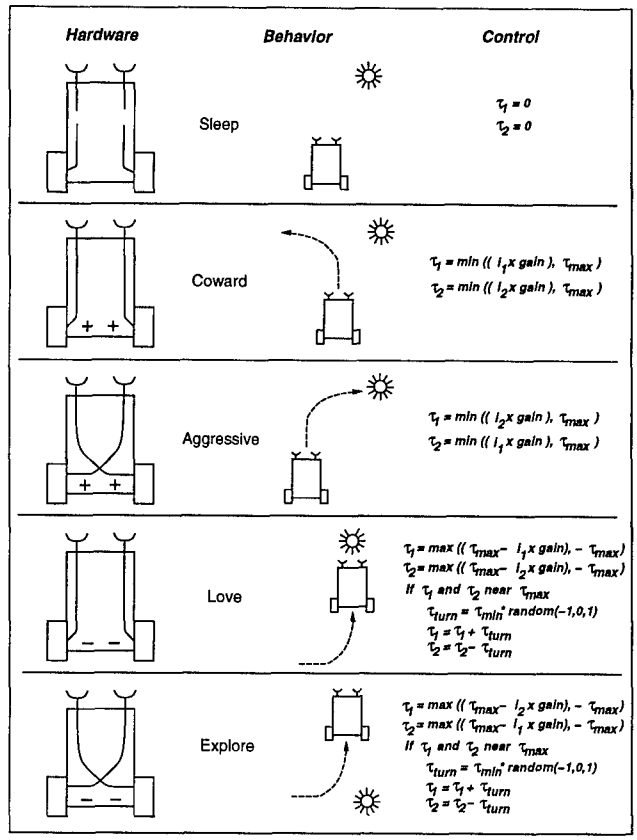


Figure 3: Description of the controllers

applies no torque to the wheels; *Coward* applies a greater positive torque ipsilaterally, turning the creature away from the light and then coming to a stop; *Aggressive*, transforms visual stimulation into contralateral motor activation, runs into the light source and, when no light is detected, comes to a stop; *Love* employs inhibitory, ipsilateral motor actuations, causing it to fix on stimuli; *Explore* employs inhibitory, contralateral motor actuations to avoid contact with any light. The learning agent is the only creature that has access to all the controllers, all the other creatures are tied to a unique controller, expressing a unique behavior.

3.3 Adaptive Control level

The adaptive control level selects which controller to activate at each moment, and computes the reinforcement signal used by the learning procedure. In addition, it receives sensory information and uses a *decoder* to transform the perceived state of the system (state of the environment plus internal state of the agent) into the corresponding learning state. Note that perceived states (or situations) usually do not describe unambiguously the true states of the system. Some states of the system are partially observable by the sensors employed. The *action selection* module selects a controller to be activated at each time and sends this information to the *learning procedure*. The evaluation function is updated over time by the *learning procedure*, determining the best controller activation command (action) for each state.

4 Simulated Environment

The environment contains a fixed number of creatures and two indicators of performance: the score which is used as the reinforcement signal, and the elapsed time since the beginning of a trial. There are up to six different types of agents: five of them assume a unique behavior, and the sixth one, the learning agent (depicted as a *frog* in Figure 4), is able to select which of the five behaviors to assume at each time. Each behavior is generated by a control law that determines the dynamics of the creature. The only way to influence the learning agent's dynamics is by sequencing controllers.

Figure 4 shows the complex environment that possesses a random set of all six kinds of creatures (*bee* - sleep behavior, *butterfly* - lover, *ant* - explorer, *roach* - coward, *beetle* - aggressive, and *frog* - learning agent). A simplified environment is shown in Figure 5 where the only creatures are *bees* and the learning agent.

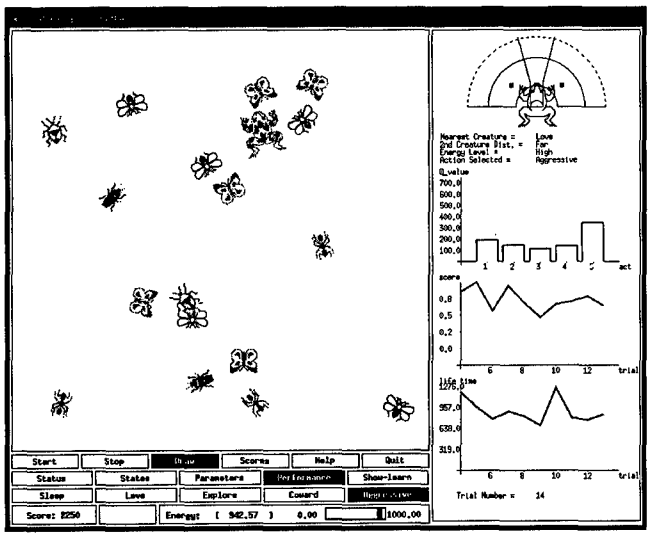


Figure 4: Complex environment

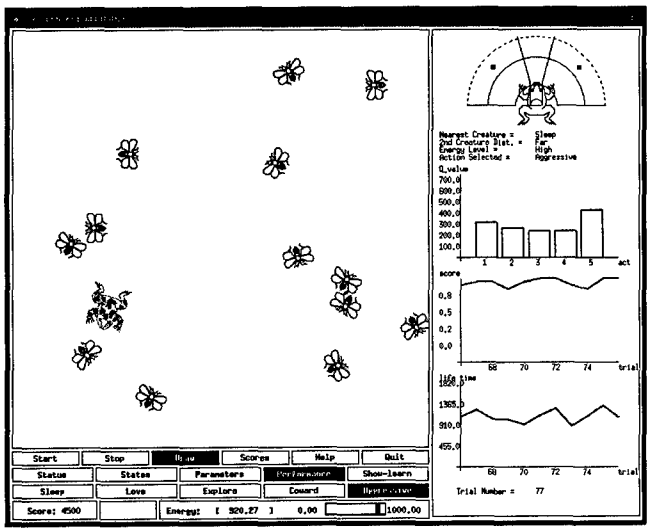


Figure 5: Simple environment

4.1 State Representation

A continuous state space can be represented by a set of discrete or continuous variables. A discrete representation permits faster learning rates and allows for a more direct interpretation of the resulting control policy, since the evaluation function is represented by a table. However the number of discrete variables increases exponentially with the problem size, becoming easily unmanageable. In the case of continuous variables, the evaluation function is represented by a function approximator (e.g. a neural network). This approach generates a compressed encoding, which is less sensitive to problem dimensionality, and induces generalization, but it may take an impracticable amount of time to be learned.

The decision of using a discrete representation was made considering the desired rate of adaptation, the fact that the control composition approach compresses the state space, and the uncertainty inherent to sensory information, forcing the use of qualitative information in discrete ranges.

When selecting a discrete representation, several issues should be considered, such as:

- Size of the state space. Unnecessary states may slow down the learning process without a significant increase in performance;
- Elimination of hidden states that degrade the learning.

The discretization of the state space used for this task is presented in Figure 6 (right). The following are the five state categories perceived by the learning agent (*frog*). The *Nearest creature distance* and the *Creature position* are represented by the distance and the angular position with respect to the learning agent. The sensor range of all agents is limited in distance, and in field of view (front 180 degrees). The dashed semi-circle in Figure 6 represents the limited range of the light sensors, and, in the case of the learning agent, also the limited range of type of creature detection. The internal semi-circles represent (starting from the innermost): the region where the center point of a prey must be in order to be considered a capture [1], *near*, *intermediate*, and *far* distances. The two central lines separates the *front* and *side* regions.

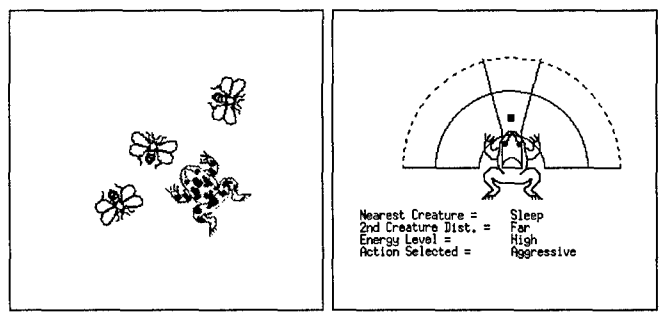


Figure 6: State representation

Usually, discretization produces ambiguous states due to deprived sensory information. Figure 7 presents a situation of a partially observable state (hidden state). These are different real states, but the agent cannot distinguish them, since they are perceived identically, causing ambiguity in the learning process. In the top example the aggressive behavior will cause the agent to miss both *prey*, simply because it is sensing similar stimuli in both light sensors and therefore moving in a straight path. In the bottom example it will result in a successful attack.

[1] As a second condition for considering a capture, the action selected by the learning agent must be *aggressive*.

State Representation:

- *Number of states: 216*
- *State Categories:*
  - Nearest Creature Type*:*
    - *(Sleep, Love, Explore, Coward, Aggressive, No Creature in Sensor Range)*
  - Nearest Creature Distance*:*
    - *(Far, Intermediate, Near)*
  - 2nd Nearest Creature Distance*:* [2]
    - *(Far, Near)*
  - Creature Position*:*
    - *(Front, Side)* [3]
  - Energy Level*:*
    - *(High, Middle, Low)*

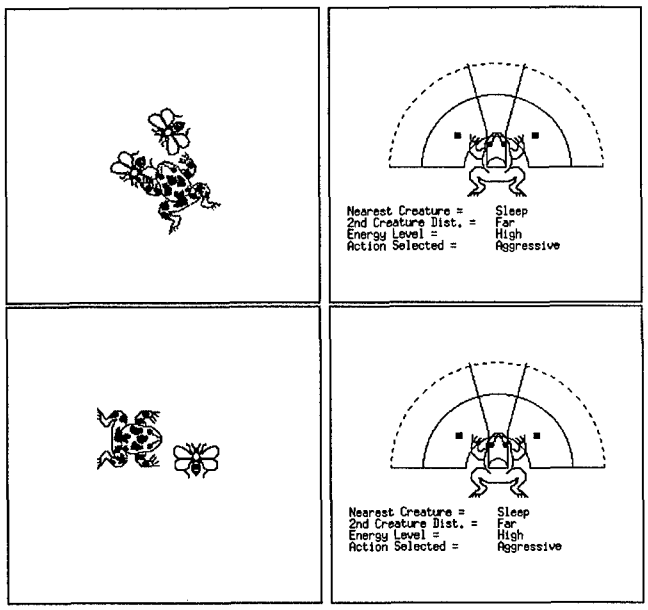


Figure 7: Hidden states

4.2 Reinforcement Signals

The selection of the reinforcement structure is also important for the performance of the learning agent, since the reinforcement and the perceived state space are the only information available to the learning process.

[3]This category was created to eliminate some hidden states that significantly degraded the learning, and it only discriminates if a second creature is nearby or not.

[3]Notice that there is no distinction between right and left side.

The learning agent receives one of the following reinforcement signals at each step of the simulation (negative reinforcement is consider a punishment and positive reinforcement, a reward for the actions performed): 750 for capturing a *Sleep* or a *Coward*; 1500 for a *Lover*; 1000 for an *Explorer*; -750 for an *Aggressive* [4]; -1 at the end of the trial; 0 otherwise. The energy level is incremented by 300, limited by the maximum energy 1000, every time that a prey is captured. Notice that the rewards are larger than the end-of-trial punishment. This structure gives priority to foraging over increasing its life-span. Since capturing prey is related to survival, the agent could, in principle, also learn foraging with only the end-of-trial reinforcement; however the sooner a good action sequence is rewarded, the faster the agent learns the task.

5 Learning Methods

Reinforcement learning is a method that does not rely on building models to attain an optimal control policy. It learns an optimal evaluation function by only receiving small reinforcements (rewards or punishments). The agent interacts with the environment, composing trial-and-error experience into reactive control rules. This paper applies two reinforcement learning algorithms to the problem of control composition: Q-learning and Q-learning with eligibility trace.

5.1 Exploration

Many reinforcement learning systems alternate between exploration and acting to improve the evaluation function. This tradeoff becomes an important issue when an optimal control policy is sought, because the amount of exploration that is necessary to find such a function is an open question. For example, the Q-learning convergence criterion to an optimal evaluation function requires each state-action pair to be tried infinitely often.

In the learning experiments presented here, exploration was attained by using a stochastic action selector that uses the Boltzmann probability distribution, which defines the probability of selecting action a in state x as:

$$p(a|x,T) = \frac{e^{Q(x,a)/T}}{\sum_{b \in A} e^{Q(x,b)/T}},$$

where T is the temperature, $Q(x,c)$ is the evaluation function for the state-action pair (x,c), for any $c \in A$, and A is the set of actions.

The percentage of exploration in a state x is determined by the temperature T and how distinct the evaluation function in this state is for different actions. Therefore, a key aspect of this method is the selection of the initial temperature value and its decay factor.

[4]After training, the learning agent will avoid the *aggressive* creatures.

5.2 Q-Learning

Q-learning is a Temporal-Difference reinforcement learning algorithm that learns an evaluation function of state-action pairs (Watkins, 1989). It is proven to asymptotically converge to an optimal evaluation function under the following conditions (Watkins, 1989; Watkins and Dayan, 1992):

- the task can be described as a Markovian decision process;
- the evaluation function is a lookup table;
- each state-action pair is tried infinitely often;
- a proper set of learning rates is selected.

These conditions are very difficult to fulfill. For example, by discretizing the input state space the second condition is fulfilled, but this usually generates hidden states, yielding a non-Markovian decision process. The cost of fulfilling both first and second conditions may be a huge state space, slowing down the learning. The third condition is subjective and dependent on the exploration procedure employed. The last condition is achieved empirically, which is, by itself, a hard task. Nevertheless, Q-learning shows normally a good performance even when these conditions are not completely satisfied. The Q-learning algorithm follows:[5]

Q-learning Algorithm:

1. *Define the current state* $\boldsymbol{x}$ *by decoding the sensory information available;*
2. *Use the stochastic action selector to determine an action* $\boldsymbol{a}$*;*
3. *Perform action* $\boldsymbol{a}$*, generating a new state* $\boldsymbol{y}$ *and a reinforcement* $\boldsymbol{r}$*;*
4. *Calculate the temporal difference error* $\hat{r}$*:*
$$\hat{r} = r + \gamma Q_{max} - Q(x, a)$$
5. *Update the Q-value of the state / action pair* $(\boldsymbol{x}, \boldsymbol{a})$*:*
$$Q(x, a) = Q(x, a) + \beta \hat{r},$$
where β *is the learning rate,* γ *is the discount factor, and*
$$Q_{max} = \max_{k \in A}(Q(y, k));$$
6. *Return to step 1.*

---

[5]This algorithm is better described in (Watkins, 1989), and is only presented here for comparison with Q-learning with eligibility trace.

5.3 Q-Learning with Eligibility Trace

The idea of using an eligibility trace in Q-learning came from its application in another Temporal Difference Method ($TD(\lambda)$) (Barto and Sutton, 1983). The main idea is to keep events "eligible" (or retain the knowledge of their occurrence), allowing for the association between those events and later ones. Therefore, if a state receives a reinforcement, all the states that contributed to its achievement get a discounted reinforcement. The objective is to accelerate the learning process in delayed reward tasks like the one presented here.

A similar algorithm ($Q(\lambda)$) that also uses eligibility trace with Q-learning was described by (Peng and Willians, 1994). An analysis of the advantages of using eligibility trace in the way described in our paper was recently published (Singh and Sutton, 1996).

The Q-learning with Eligibility Trace algorithm operates as follows:

Q-learning with Eligibility Trace Algorithm:

1. *Define the current state* $\boldsymbol{x}$ *by decoding the sensory information available;*
2. *Use the stochastic action selector to determine an action* $\boldsymbol{a}$*;*
3. *Perform action* $\boldsymbol{a}$*, generation a new state* $\boldsymbol{y}$ *and a reinforcement* $\boldsymbol{r}$*;*
4. *If* $(a \neq arg\ max_{b \in A}(Q(x, b)))$*, erase the eligibility trace by setting all* $e(x_i, a_i)$ *to 0;*
5. *Calculate the temporal difference error* $\hat{r}$*:*
$$\hat{r} = r + \gamma Q_{max} - Q(x, a)$$
6. *Set* $e(x, a)$ *to* 1.0*;*
7. *Update all* $Q(x_i, a_i)$ *and* $e(x_i, a_i)$ *values:*
$$Q(x_i, a_i) = Q(x_i, a_i) + \beta \hat{r} e(x_i, a_i)$$
$$e(x_i, a_i) = \alpha e(x_i, a_i),$$
where β *is the learning rate,* γ *is the discount factor,* α *is the eligibility factor, and*
$$Q_{max} = \max_{k \in A}(Q(y, k));$$
8. *Return to step 1.*

The eligibility trace is erased if the action that will be performed in the new state is not the one associated with Q_{max}. This is required to maintain coherence, avoiding to follow non-optimal policies (Lin, 1993).

5.4 Experiments

All the experiments presented in this section are related to the simple environment (Figure 5) where only one

type of prey, representing the sleep behavior, is present. Figure 8 shows the learning performance of Q-learning and Q-learning with eligibility trace. In these graphs, each point represents the sample mean of scores obtained over the same set of 30 random trials using the correspondent evaluation function (without exploration). The error bars represent the 95% confidence intervals for the scores' population means. These graphs show the two most distinct learning curves (*Run 1* and *Run 2*) out of 10 runs. For both algorithms the runs differ in the random sequence used at each trial. Q-learning exhibited the lower-type performance (*Run 2*) for 70% of the runs, converging on average to a sub-optimal evaluation function. On the other side, Q-learning with eligibility trace always converged to a near-optimal evaluation function, outperforming Q-learning on this problem, under the learning parameter setting described below.

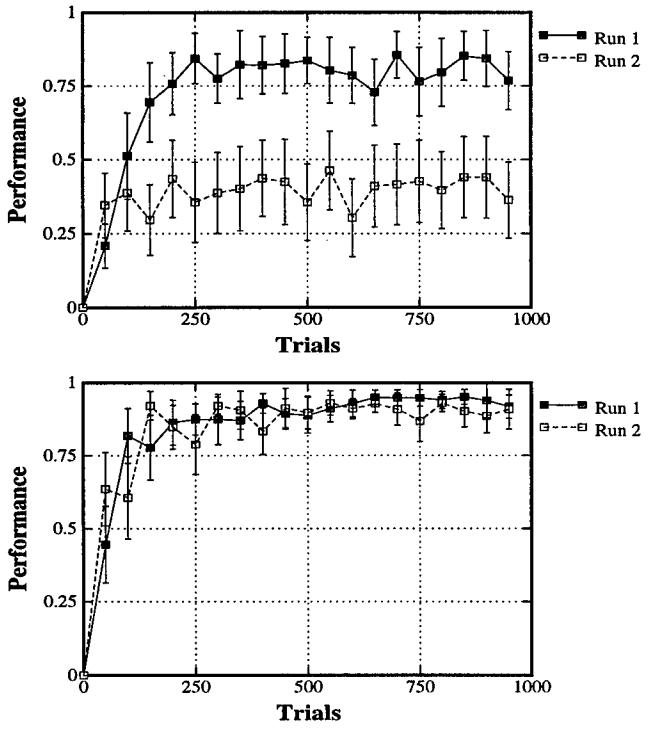


Figure 8: Learning curves using Q-learning (top) and Q-learning eligibility trace (bottom)

From the runs depicted in Figure 8, the evaluation functions that produced the best average performance on each run were saved. Their performance distributions over a set of 300 random trials are shown in Figures 9 and 10. Again, this test is performed on fixed state-action tables generated by the evaluation functions - no exploration is allowed during the test. Figure 11 shows a comparison between a state-action table produced by Q-learning with eligibility trace and a carefully handcrafted one. This indicates that the solutions found by this learning algorithm are near-optimal, although not significantly different from the handcrafted solution.

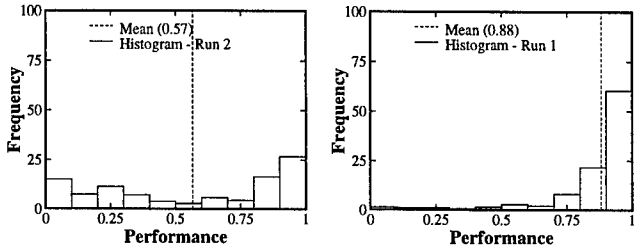


Figure 9: Q-learning performance histograms

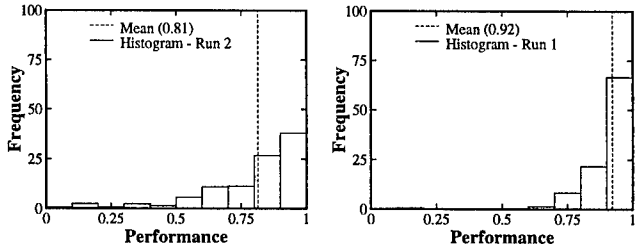


Figure 10: Q-learning eligibility trace histograms

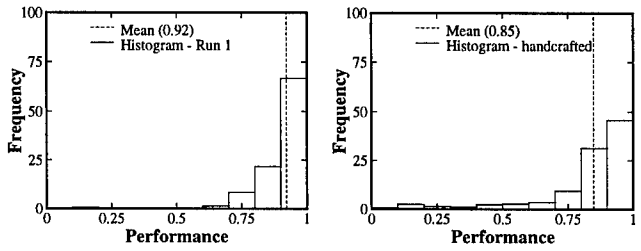


Figure 11: Q-learning eligibility trace (left) and a handcrafted state-action table (right) histograms

Learning Parameters:

- *learning rate* ($\beta = 0.025$);
- *discount factor* ($\gamma = 0.95$);
- *eligibility factor* ($\alpha = 0.95$);
- *initial temperature* = 2000.0;
- *final temperature* = 1.0;
- *temperature decay factor* = 0.98;
- *Q-values and eligibility trace were initialized with* 0.0;
- *The state-action table is updated at each step of the simulation.*

All the parameters were empirically selected. The exploration level does not decay completely after reaching the final temperature. An average of 10% of all the actions at the end are still caused by exploration. This happens because some states do not have a definite best action, due to hidden states.

6 Learning in a Complex Environment

After testing the above learning methods with the simple environment, the next step is to learn the complex environment (all the six types of creatures present). This environment presents an additional challenge, since the new four types of creatures continuously disturb the environment with their dynamics. The chance of capturing a prey by chance in this new situation is minimal, since the new types of creature (*love*, *coward*, and *explore*) are tuned to react to other creatures, or represent a negative reinforcement (poison) when captured (*aggressive*).

As depicted by the learning curve in Figure 12 (top), learning on this new environment is a harder task, the best performance produced by the Q-learning with eligibility trace does not reach the 75% level.

Shaping is known in experimental psychology and reinforcement learning as a method for accelerating the learning process (Gullapalli and Barto, 1992), and in extreme cases, making learning of complex behaviors possible. Shaping methods divide the learning task in stages; initially, reinforcement is given whenever a set of states containing the goal state is reached. As the agent reaches proficiency in each stage, it moves to the next stage, in which the reinforcement structure is refined, being applied only to states closer to the goal state. This process continues until the complex task is mastered.

The shaping approach used here manipulates the dynamics of the creatures (prey) over time. The idea is to start with almost static prey (high impairment) that are easy to capture, increasing the chance of reinforcement. Over the learning process, the impairment is reduced, letting the dynamic effects increase slowly. Appendix A presents a description of the dynamics of the creatures. The shaping procedure used is described below:

Shaping Procedure:

1. *Set impairment coefficient k ($k \in \{10, 5, 2\}$) to the maximum value ($k = 10$);*
2. *Set learning exploration to its maximum;*
3. *Learn new shaping stage on top of the evaluation function previously learned, until a performance index of* 70% *or* 500 *trials are reached;*
4. *If not in the last shaping stage ($k \neq 2$), set impairment coefficient to next value, and return to 2.*

Figure 12 compares the learning curves obtained by Q-learning with the shaping procedure (bottom) and without (top). As shown, the maximum performance of the learning algorithm without shaping is on average lower than 75%, while shaping produces an average performance above this limit. All curves presented here were obtained using the same experimental procedure as in Section 5.4, except that only 3 runs where performed.

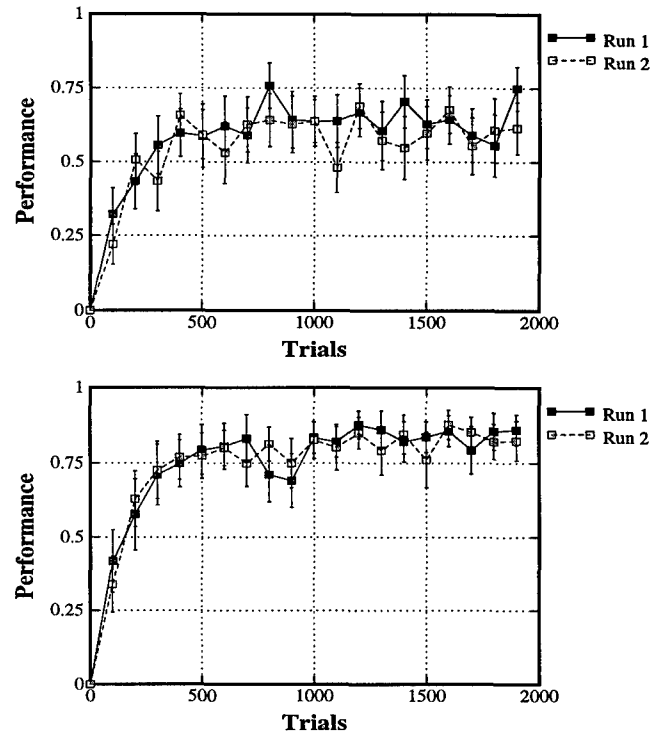


Figure 12: Learning curves using Q-learning with eligibility trace without (top) and with shaping (bottom)

Figures 13 and 14 show the improvement in performance given by the shaping procedure. In Figure 15, there is a comparison between the evaluation function created by the learning algorithm with shaping, and a carefully handcrafted version, indicating that the learning algorithm converged to a near-optimal solution.

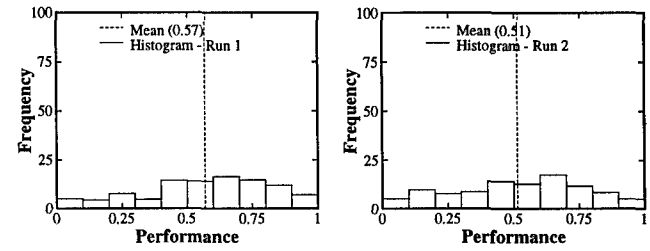


Figure 13: Performance histograms for learning without shaping

7 Conclusion and Future Work

In the simple environment, both learning methods presented interesting results. The discrete representation seems to generate too many hidden states, degrading the performance of the Q-learning algorithm. However,

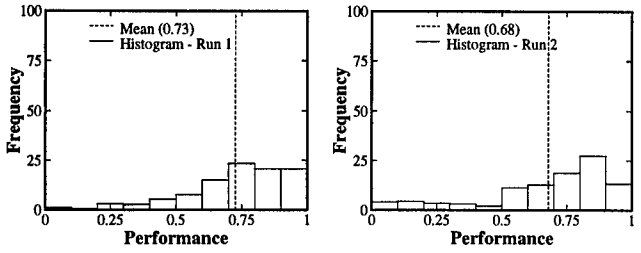


Figure 14: Histograms for learning with shaping

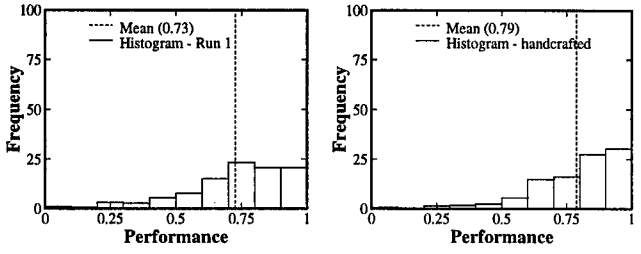


Figure 15: Histogram for shaping (left) and a hand-crafted state-action table (right)

Q-learning with eligibility trace seems to be less sensitive to these problems, performing similarly to our best guess for state-action tables. The complex environment is extremely dynamic, increasing even more the occurrence of hidden states. In this situation, the introduction of shaping improves the final performance of the learning algorithm.

The complexity of the problem presented was reduced by at least two orders of magnitude in state space size by expressing it as a control composition task. A set of very simple controllers and a simple learning procedure were the final requirements for solving this simulated foraging task. Control composition can be understood itself as a shaping procedure, where the agent acquires complex behaviors by composing relevant behaviors.

Research in control composition is starting and while this new paradigm shows some promise, it also creates a new set of challenges, such as how to design a control basis able to span the control complexity desired, how to compose adaptive controllers, and how to generalize control strategies over tasks.

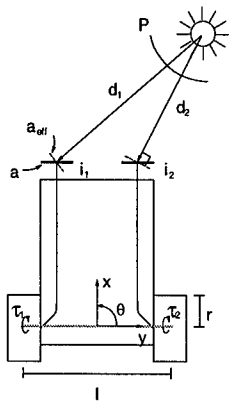
Acknowledgments

The research reported here was conducted in the Laboratory for Perceptual Robotics at UMASS and supported in part by NSF CDA-8922572, IRI-9116297, and IRI-9208920.

The authors would like to acknowledge the contribution of Manfred Huber especially in the design of Q-learning with eligibility trace, the comments of Prof. Paul Utgoff from an early version of this paper presented at a machine learning seminar in 1993, and to Sergio Gúzman-Lara for his advice on the experimental part of this paper.

A Simulation Dynamics

Each creature is modeled as a vehicle with two wheels, as in Figure 16.



Creature Parameters:

m	*Mass;*
I	*Moment of inertia;*
r	*Wheel radius;*
l	*Axle length;*
i_1, i_2	*Light intensity captured by left and right sensors;*
d_1, d_2	*Distances from light source to left and right sensors;*
τ_1, τ_2	*Torque applied to the left and right wheels;*
P	*Power irradiated;*
a	*Light sensor area;*
a_{eff}	*Effective light sensor area;*

Figure 16: Creature model

The following formulas approximate the dynamics of the vehicle. The velocity v was derived considering constant torque between time steps:

$$v_t = v_{t-\Delta t} + \int_{t-\Delta t}^{t} \left(\frac{F}{m} - \frac{D}{m} v_t\right) dt \qquad (1)$$

$$\bar{a}_t = \frac{v_t - v_{t-\Delta t}}{2}$$

$$s_t = s_{t-\Delta t} + v_{t-\Delta t}\Delta t + \frac{1}{2} a_t (\Delta t)^2,$$

where v_t is the velocity, $\bar{a}_t$ is the average acceleration, and s_t is the position, at time t. F is the force applied to the center of mass of the vehicle, and D is the damping factor.

Solving Equation 1 for the 3 degrees of freedom of the vehicle (x, y, θ), results in:

$$\dot{x}_t = \left(\dot{x}_{t-\Delta t} - \frac{F cos(\theta_{t-\Delta t})}{D_{tr}}\right) e^{\frac{-D_{tr}}{m}\Delta t} + \frac{F cos(\theta_{t-\Delta t})}{D_{tr}}$$

$$\dot{y}_t = \left(\dot{y}_{t-\Delta t} - \frac{F sin(\theta_{t-\Delta t})}{D_{tr}}\right) e^{\frac{-D_{tr}}{m}\Delta t} + \frac{F sin(\theta_{t-\Delta t})}{D_{tr}}$$

$$\dot{\theta}_t = \left(\dot{\theta}_{t-\Delta t} - \frac{\tau}{D_{rot}}\right) e^{\frac{-D_{rot}}{I}\Delta t} + \frac{\tau}{D_{rot}},$$

where D_{tr} and D_{rot} is the translational and rotational damping, respectively.

The force F and torque τ are given by:

$$F = \frac{\tau_1 + \tau_2}{2kr} \qquad \tau = \left(\frac{\tau_1 - \tau 2}{kr}\right) \frac{l}{2},$$

where k is the impairment coefficient (1 for the learning agent, and 2 for the other creatures).

The light intensity ($i_j(k)$), collected by sensor j of vehicle k from all other vehicle in the environment($s \neq k$), is computed by the formula below:

$$i_j(k) = \sum_{s \in S, s \neq k} \left(\frac{P}{4\pi\, d(s,k_j)^2} a_{eff}(s,k_j) \right),$$

where $a_{eff}(s,k_j)$ is the effective area of incidence for the energy irradiated from vehicle s onto sensor j in vehicle k, $d(s,k_j)$ is the distance from vehicle s to sensor j of vehicle k, and S is the set of all vehicles in the environment which are in sensor range R, and within the front semi-circle. Also, if $d(s,k_j)$ is less than a minimum distance d_{min}, the distance is set to d_{min}. For details, see diagram on Figure 16.

The energy (e) of the learning agent is incremented by 300 every time that a creature is captured, and decreased at each step of the simulation in a rate that varies from 1 to 2 units, depending on its translational velocity, as described in the formula below:

$$e_t = e_{t-\Delta t} - 1 - \frac{\sqrt{\dot{x}^2_{t-\Delta t} + \dot{y}^2_{t-\Delta t}}}{V_{max}}$$

$$V_{max} = \frac{F}{D_{tr}},$$

where V_{max} is the maximum translational velocity.

References

Barto, A. G. and Sutton, R. S. (1983). Neural problem solving. Technical Report 83-03, Department of Computer Science, University of Massachusetts, Amherst.

Barto, A. G., Sutton, R. S., and Watkins, C. J. C. H. (1990). Learning and sequential decision making. In Gabriel, M. and Moore, J., editors, *Learning and Computational Neuroscience: Foundations of Adaptive Networks*, chapter 13, pages 539–602. M.I.T. Press.

Braitenberg, V. (1984). *Vehicles - Experiments in Synthetic Psychology.* M.I.T. Press, Cambridge, MA.

Grupen, R., Huber, M., Coelho Jr., J. A., and Souccar, K. (1995). Distributed control representation for manipulation tasks. *IEEE Expert, Special Track on Intelligent Robotic Systems*, 10(2):9–14.

Gullapalli, V. and Barto, A. G. (1992). Shaping as a method for accelerating reinforcement learning. In *Proceedings of the 1992 IEEE International Symposium on Intelligent Control*, pages 554–559, Glasgow, Scotland, UK. IEEE.

Huber, M., MacDonald, W. S., and Grupen, R. A. (1996). A control basis for multilegged walking. In *Proceedings of the International Conference on Robotics and Automation*, volume 4, pages 2988–2993, Minneapolis, MN. IEEE.

Lin, L.-J. (1993). *Reinforcement Learning for Robots using Neural Networks.* PhD thesis, Carnegie Mellon University, Pittsburgh, PA.

Maes, P. and Brooks, R. A. (1990). Learning to coordinate behaviors. In *Proceedings Eighth Nacional Conference on Artificial Intelligence*, volume 2, pages 796–802. AAAI Press / The MIT Press.

Mahadevan, S. and Connell, J. (1990). Automatic programming of behavior-based robots using reinforcement learning. Technical report, IBM Research Division, T. J. Watson Research Center, Yorktown Heights, NY 10598.

Peng, J. and Willians, R. J. (1994). Incremental multi-step q-learning. In Cohen, W. W. and Hirsh, H., editors, *Machine Learning: Proceedings of the eleventh International Conference*, pages 226–232, New Brunswick, NJ. Morgan Kauffmann Publishers.

Singh, S., Barto, A., Grupen, R., and Connolly, C. (1994). Robust reinforcement learning in motion planning. In *Advances in Neural Information Processing Systems 6*, pages 655–662. Morgan Kaufmann Publishers.

Singh, S. P. and Sutton, R. S. (1996). Reinforcement learning with eligibility traces. *Machine Learning - Special Issue on Reinforcement Learning*, 22(1/2/3):123–158.

Sutton, R. S. (1990). Reinforcement learning architectures for animats. In *From Animals to Animats : Proceedings of the First International Conference on Simulation of Adaptive Behavior*, pages 288–296, Paris, France. M.I.T. Press.

Sutton, R. S. and Barto, A. G. (1990). Time-derivative models of pavlovian reinforcement. In Gabriel, M. and Moore, J., editors, *Learning and Computational Neuroscience: Foundations of Adaptive Networks*, chapter 12, pages 497–537. M.I.T. Press.

Watkins, C. J. C. H. (1989). *Learning from Delayed Rewards.* PhD thesis, Cambridge University, Cambridge, England.

Watkins, C. J. C. H. and Dayan, P. (1992). Techinal note: Q-learning. *Machine Learning - Special Issue on Reinforcement Learning*, 8(3/4):279–292.

Modular growing network architectures for TD learning

Pascal BLANCHET
CRIN-CNRS / INRIA Lorraine
BP 239, 54506 Vandœuvre-Lès-Nancy, France
E-mail: blanchet@loria.fr

Abstract

We present three variants of a modular connectionist architecture for learning behaviors with delayed rewards by TD methods. This architecture is based on connectionist modules that are dedicated to different parts of a given task. The aim of this work is to explore the possibilities of such systems to learn explicit internal representations of their behaviors. The idea is that the more explicit is the learned representation, the more easily it could be reused in new situations. The difficulty is to handle the constraint of building an explicit representation without decreasing the robustness of the TD algorithm. In this paper, we propose two different growing network algorithms that build explicit (or partly explicit) representations, and we compare their performances to a more classical architecture. The experiments have been done in a simulated environment containing some real world data (hand-written digits) in order to evaluate their robustness to a pattern recognition subproblem.

1 Introduction

One of the questions which motivate our work is how an artificial agent could learn as autonomously as a human being. More precisely, the question is what kind of algorithm could learn almost alone very different capabilities that are usually performed by different, and often incompatible, AI systems. The learning paradigm which seems the most appropriate to this question is reinforcement learning with delayed reward. In this approach the agent must learn sequences of actions by trial and error, in order to maximize the quantity of reward received from the environment. The difficulty is that the system does not get an evaluation of the fitness of its decision after each trial. In the best case it receives a reward after a sequence of actions, once it has reached a goal situation for example. The temporal difference algorithms (TD) [16] can solve this problem. It has been successfully applied to a lot of different tasks, such as backgammon playing [17] or mobile robot control [12].

The main problem that has to face a TD algorithm occurs when the probability to get a reward by random exploration of the environment is to low. In such a case, the rewards received by the systems are too rare, its actions are negatively evaluated most of the time, and no sequence of actions can be learned. This problem has been addressed through many different approaches. Some of them have also been combined with success in single systems [10].

The first evident solution consists in teaching the agent [10]. In this case, the good sequences of actions are explicitly given. The learning algorithm is the same, it is only the action generation process which is changed. But this method requires the intervention of a human expert, and this is what we want to minimize.

Another interesting solution consists in dividing the whole task into subtasks that can be solved by random exploration easier [11] [12] [7] [8]. Once the system has learned to perform several subtasks, these subtasks can be used at a higher level as macro-actions. Then, if the probability to solve the whole task by randomly trying these new macro-actions is high enough, the system will have a good chance to learn it. This solution requires more efforts from the human designer. But this knowledge is less precise than what a teacher must give.

Some researchers have also proposed planning technics [10] [15]. Planning consists in using a world model to predict the effect of a sequence of actions. No external intervention is required if the system is able to learn its own world model from its experience. But, like in classical AI, planning can be very costly in some situations. The number of possibilities to explore before generating one plan may sometimes be very large.

Therefore, in addition to all these technics, we think that the system should also be able to explicitly memorize the sequences of actions and perceptions that it uses to solve its problems. The idea is that if the sequences are more precisely represented in memory, when the system faces a new problem, it can "remember" what it has previously done in a different but similar situation and try to do the same thing or adapt its previous solution to the new context. The adaptation is often a matter of analogy or a matter of different variable setting in a general frame. This is a classical problem in symbolic

AI. The difficulty is to access to these technics within the frame of TD learning in dynamic and noisy environments.

The goal of all these methods is to generate better sequences of actions than the simple random exploration process. The advantage of TD learning is that if a bad sequence of actions is generated, the learning algorithm will not keep it. It is not necessary to find complex algorithms that generate perfect solutions. Thus, we think that there should exist many methods, more or less inspired from symbolic AI, that could help to propose solutions from the knowledge previously acquired by a TD learning agent. It seems clear that the best way to have a learning system which boot-straps toward a high performance level, is to have a collection of different powerful methods that can reuse what the agent has already learned.

The problem of our approach [3] [4] is that explicit representations are known to be brittle to noisy real world data. Moreover, building and evaluating a representation is more complex than simply evaluating the utility of an action, all the more since the system will have to handle both tasks at the same time. The question is what kind of algorithm and what kind of representation will be able to satisfy these requirements with a learning speed and a robustness similar to classical TD learning algorithms.

In this paper, we present a modular TD learning connectionist architecture which has been implemented with three different kinds of modules. The first kind of module uses a classical neural TD learning algorithm, which does not build any representation. The two other ones are growing networks which build tree-like structures more or less explicitly representing the sequences of actions and perceptions that lead to rewards.

The systems have been tested in a simulated environment with additional real world data to check the robustness of each algorithm. The task consists in learning to manipulate a block on which a hand-written digit is displayed. The agent must learn to move the block from its initial place to a target place which depends on the displayed hand-written digit.

In the following sections, we describe the modular organization which is common to our three systems. We then sketch the three learning algorithms. Finally we present the experiments and discuss the results that we obtained with the different algorithms.

2 The modular TD learning agent

The agent has a set of feature detectors providing real values between 0 and 1. It has a set of discrete output effectors (binary output units), to modify the state of its environment. The task that the agent must learn is characterized by a final goal situation which can be well detected by one or several feature detectors. It receives a reward only when this goal situation is achieved. The architecture contains several interconnected modules. Each module has the same functionalities as the global system, except that its goals are more simple, and easier to achieve.

3 The TD learning modules

A module is a kind of small agent which can learn parts of the complete agent's task. It must be able to learn one or several different subtasks more or less useful for the complete task. Each subtask T_i of a module must be characterized by a goal situation G_i. A module M is characterized by the three following properties given by the designer of the agent:

- a set of input units : $IN(M)$
- a set of output units : $OUT(M)$
- a set of goal units : $GU(M)$

A module M does-not need to be connected to every input or output units. It is important to select which parts of the input and output data are relevant to the tasks assigned to the module. Reducing the input/output connections can provide *a priori* generalization, and can focus the random exploration on the *a priori* useful actions.

A goal unit can take two activity values: 0 or 1. Each goal unit $g_i \in GU(M)$ corresponds to one of the subtasks of the module M. The weights of the connections between a goal unit g_i and the input units are manually set by the designer, so that g_i is activated if and only if the goal situation G_i is achieved.

A module M can do (or learn to do) only one of its assigned subtasks T_i at a time. When M is trying to do the task T_i, we say that its goal G_i is selected. When a module has no selected goal, it is inactive. When G_i is selected, the activation value of the unit g_i is considered as the reward of the module. At this time, the module behaves (and learns to behave) in order to maximize the average activation of g_i over time.

3.1 Connections between modules

The modules are hierarchically connected together so that some of them can use the capabilities of the others to solve their own goals. A module can use another one simply by selecting one of its goals. The selection of a given goal of a given module is an action, as well as the external effectors. There exists one effector gs_i for each goal G_i of a module M. When gs_i is triggered by another module, the goal G_i is selected and M behaves in order to achieve it.

For simplicity, the connections between modules are full or empty. If a module M_1 can select one goal G_i of a module M_2 ($gs_i \in OUT(M_1)$) then it can also select any

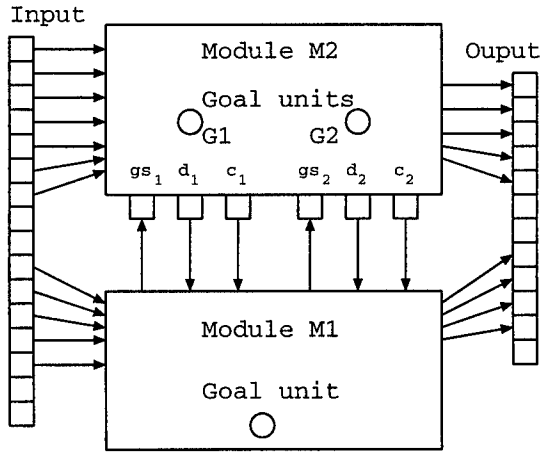


Figure 1: *Connection between two modules (M_1 controls M_2).*

goal of M_2, and we say that M_1 controls M_2. We have not allowed partial connections nor bidirectional connections (i.e. if M_1 controls M_2 then M_2 cannot control M_1).

Since the control of a module may require some information about the internal state of this one, we have associated two feature detectors d_i and c_i to each goal G_i of each module. d_i is 1 if the activity of the goal unit g_i is above a fixed threshold (0.9 in our experiments) and 0 otherwise. c_i is 1 if the module "knows" how to achieve the goal G_i, and 0 otherwise. c_i roughly means that the necessary preconditions for achieving G_i are satisfied (this notion will be explained later). The figure 1 shows how two modules can be connected together and to the global input/output device.

Finally, the agent contains one top level module. Its goals are selected by an external cause (the designer in our case). This unique top level module can control other modules which may control other lower level modules etc.

4 Neural TD algorithm

4.1 Principle

In this variant, the modules use a Q-learning algorithm implemented by formal neurons [11] [16]. If we call X the set of input states of the module M, then M will have to learn a function $Q(x, a)$ which predicts the future amount of reward that the module should receive if it uses the action $a \in OUT(M)$ when $x \in X$ is perceived by M.

Since a module must learn different subtasks T_i rewarded by the activity of different goal units g_i, it needs one function $Q_i(x, a)$ for each subtask. We have implemented the Q_i function by formal neurons. There is one neuron $n(i, a)$ for each action $a \in OUT(M)$ and each goal G_i. If $input(t)$ is the state presented to the module at time t, then the activity of the neuron $n(i, a)$ at this time ($Act(n(i, a), t)$) corresponds to $Q_i(input(t), a)$. In the following, we call N_i the set of units involved in a given goal G_i, this set contains g_i and any unit $n(i, a)$ with $a \in OUT(M)$.

When running, a module whose goal G_i is selected, just has to select the action associated to the best activated unit in N_i at each step. It can also select a random action in order to explore different possibilities.

4.2 Particularity of goal selection actions

Since a goal selection action is a macro action, the time necessary for its execution is longer than the time necessary for the execution of a simple external action (one cycle). A goal selection action is completed when the corresponding goal is achieved. Therefore, if a module such as M_1 (in figure 1) execute the action gs_1 of M_2, it must wait for the achievement of G_1 before using another action. But on the other hand, if M_2 cannot achieve G_1, M_1 must not wait for ever. Thus, an execution time limit (100 or 200 cycles) has been associated to each goal selection action. When this limit is reached before the achievement of G_1, M_1 can try another action. In fact, this situation mainly occurs at the beginning of learning, when the modules are learning how to solve their goals. This time limit can be considered as the maximal duration of an exploration phase. This is why they must be long enough.

A problem occurs when a better solution appears during the execution of an action. The module must decide whether it continues its current action or whether it tries the new appealing solution. This could be implemented by choosing the action of another unit if it has a higher activity than the activity that had the unit responsible for the execution of the current action when the decision was taken.

But we have observed that this method entails too many action changes so that the exploration time of the lower level modules is not long enough to learn. Therefore, a new action will be chosen if the neuron which proposes it has an activity value above a significant threshold (0.9 for example).

4.3 Decision cycle

We describe the method used by a module M to select an action. This method requires that the activity of the neuron which is responsible for the choice of an action at a given time is memorized for future comparison. We call $t_{decision}$, the last instant when M started the execution of a new action. Here is the step by step method followed by a module to choose an action at time t:

1. Compute the activity of each unit of M.
2. For each goal G_j, find $n_{best}(j, t)$, the unit of N_j which has the greatest activity value at time t.

3. If there is no selected goal in M or if the selected goal is already achieved, then do nothing. Otherwise go to next step.
4. Let G_i be the selected goal. If G_i has just been selected then go directly to step 6, else go to next step.
5. If the activity $n_{best}(i,t)$ is below 0.9 or below $Act(n_{best}(i,t_{decision}),t_{decision})$, and if the current action is not completed, then continue the current action. Otherwise, go to next step.
6. Use the action of $n_{best}(i,t)$ or a random action with a probability roughly equal to the activity $n_{best}(i,t)$. Affect t to $t_{decision}$.

4.4 Activation of c_i

The internal feature detector c_i of the goal G_i is activated at time t if $Act(n_{best}(i,t),t) \geq 0.9$. When a unit is activated with a so high value, it generally means that the system is able to achieve G_i from the current situation.

4.5 Learning algorithm

If the current selected goal at time t is G_i, and if the action a is just completed, then the input weights of the unit $n(i,a)$ are updated according to the following formula :

$$w_{j,n(i,a)} = w_{j,n(i,a)} + error(t) \ . \ input_j(t_{decision}) \ . \ \varepsilon$$

where $w_{j,n(i,a)}$ is the weight between the input unit j and the neuron $n(i,a)$, $input_j(t)$ is the input value number j at time t, ε is the learning coefficient (typically 0.1), and $error(t)$ is the prediction error defined as follows :

$$\gamma \times (Max_{n \in N_i}(Act(n,t)) - Act(n(i,a),t_{decision}))$$

where γ is a discount coefficient that we set to 0.97 to obtain a good convergence rate (with 0.99 the system did not correctly learn). With γ, the module will learn the difference between situations far from the goal and situations close to it.

5 the first growing network algorithm

In this second system, the units are dynamically created and organized in a tree-like structure. The root of the tree is the goal unit, while the nodes are formal neurons which perform state to action associations (figure 2). Each sequence of units, from a given node to the goal unit, represents a sequence of actions from an initial state to a goal state.

In this scheme, each unit (except the goal unit) is associated to another unit which is called its successor (it can be the goal unit). For example, in figure 2, the successor of $u3$ is $u2$ and the one of $u1$ is g_i. Here, the meaning of $Act(u3,t)$ is different than in the previous system. It represents an estimation of $Act(u2,t')$ assuming that a is executed at time t and completed at time t'. More clearly, the activity of a unit predicts the future activity that its successor will have after the execution of its action. If the estimation is good for every unit of the tree, it means that when a unit such as $u3$ has its activity above 0.9 then the goal unit g_i will have a similar activity if the actions associated to $u3$, $u2$ and $u1$ are successively executed.

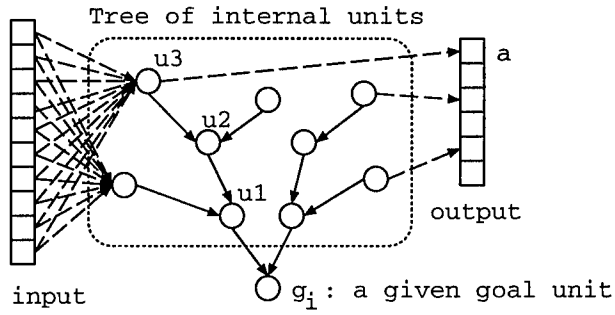


Figure 2: *A goal unit and its associated tree of units. Each internal unit is connected to every input units, and to one output unit. (For clarity, only a part of the input and output connections has been displayed.)*

5.1 Learning algorithm

The decision cycle of the growing networks is the same, but the learning algorithm needs to be adapted. The input weights of a unit can be learned with an error value representing the difference between its activity at the decision time and the one of its successor at time t when its action is completed :

$$error(t) = Act(u2,t) - Act(u3,t_{decision})$$

The γ coefficient is no more useful since the distance to the goal is given by the number of units between $u3$ and the goal unit. This is rather important since the previous system is very sensible to a modification of γ. (We also found that better results could be obtained by a network which uses only the sign of this error. The growing network is more robust to parameter setting problems.)

In this system, a unit like $u3$ is adapted only if it was the unit responsible for the decision to use its action a at time $t_{decision}$. This is important since several units can be associated to the same action a, and we have found in our experiments, that it was never good to adapt a unit which is not responsible for the choice of the action. Therefore, it is necessary to adapt the good one.

The learning algorithm also needs a criterion to add new units to the tree. A new unit such as $u3$ will be associated to action a and to the successor $u2$ at time t, if each one of the following conditions is satisfied:

- $u2 = n_{best}(i,t)$ and $Act(u2,t) > 0.9$

- the action a has just been completed
- there does not already exists another unit u associated to a and $u2$.

The input weights of the new created unit are set to 0.

5.2 *The problem of explicit representation*

This architecture is more constrained than the previous one. It is the normal consequence of an explicit internal representation. The main problem is that a sequence of units represents a fixed sequence of actions from an initial situation to a goal situation. As a matter of fact, let us consider the figure 2 in the following situation: $u3$ is $n_{best}(i, t_{decision})$. Then suppose that after the completion of a at time t, $Act(u2, t) < Act(u3, t_{decision})$ but $Act(u1, t) > Act(u3, t_{decision})$. In such a case, it is clear that in regard to the current goal G_i, the use of the action a is a success. But if we only consider the ability of $u3$ to predict the activity of $u2$ when a is used, it is rather a failure.

In other words, this system is less adapted to learning actions that only have a global good effect. It is made for learning actions that make identifiable steps toward identifiable goals.

In order to solve this problem, we consider that the weights of a unit such as $u3$ will be updated at time t (when a is completed), only if one of the following conditions is satisfied by its successor $u2$:

- $Act(u2, t) > Act(u3, t_{decision})$.
- There is no other unit u closer to the goal than $u3$, which satisfies $Act(u, t) > Act(u3, t_{decision})$.

Another related problem occurs when some actions need to be repeated several times to produce a significant effect. For example, when someone tries to start the engine of his car, he never knows how much time he will have to turn the contact key before having the engine started.

This variability in the time dimension must be explicitly taken into account in this architecture. The problem is that when the action a associated to $u3$ is in this category of actions, then the system will consider as a failure each execution of a which is not immediately followed by the desired effect, the activation of $u2$. The task will be impossible to learn if there is no clue to enable the system to exactly know how long a must be executed.

The solution we have used consists in associating to each action a maximal number of repetition that must be done before considering a possible failure. For example, if the engine of the car does not start after five trials then there must be an important problem. This delay is different from the previous one, which only concerned the time necessary for the completion of the action. The present delay concerns the number of completion of the action necessary to get the desired effect.

A similar solution has been used by Benson [1] in the system TRAIL. The difference is that in our case, the action can go on even if its condition is no more satisfied (i.e. even if the activity of the unit falls down). Finally, let us notice that this problem was implicitly treated by our previous architecture.

6 The second growing network algorithm

This method is close to the one described in [3]. It builds a tree similar to the previous growing network, but the units are not formal neurons. In the previous networks, the weights of the units are adapted in order to learn the utility values of the actions. In this case, the weights of the units are fixed and the utility of each unit is a coefficient which is computed during learning.

The learning mechanism can be compared to the one used in Holland's classifier system [5]: it generates units, test them, rewards the good ones, and removes the less useful ones.

In this system, each unit n is dedicated to the recognition of a fixed set of input states (let us call this set $I(n)$). n contains a utility coefficient ($utility(n)$) which represents the estimated probability to reach the goal situation assuming that the actions associated to n and to the units on the pathway from n to the goal unit, are successively executed from an instant t such that $input(t) \in I(n)$. Here is how the activity is computed:

$$Act(n, t) = (input(t) \in I(n)) \times utility(n)$$

where $input(t) \in I(n)$ if and only if:

$$(\sum_i Min(input_i(t), w_{i,n})) \geq (\alpha \sum_i w_{i,n})$$

where α was 1 in our experiments. This formula means that each weight is an activity threshold of the corresponding input unit. If α is 1, n is activated only if each input value is greater or equal to its corresponding weight (or threshold). When $\alpha < 1$, a small variation below the thresholds is accepted.

The advantage of this solution is that the meaning of each weight of each unit is clear. The set of weights associated to an internal unit represents an input prototype which could be easily analyzed, compared or combined with other ones. This is important for an evolution toward symbolic capabilities.

6.1 *The learning algorithm*

6.1.1 *Learning the utility*

Learning occurs at the same time it occurs in the first growing network. But instead of adapting the weights,

we adapt only one coefficient per unit such as $u3$: its efficiency $eff(u3)$. This coefficient estimates the probability that the successor $u2$ of $u3$ will be activated thanks to the execution of a. $eff(u3)$ is modified according to this formula:

$$eff(u3) = eff(u3) + \varepsilon \times success(u2,t) \times (1 - eff(u3))$$

where $success(u2,t)$ is -1 if $Act(u2,t)$ is 0 and 1 if $Act(u2,t)$ is positive (which only occurs when $input(t) \in I(u2)$). ε was set to 0.1.

The utility of $u3$ is deduced from its efficiency and the utility of its successor $u2$:

$$utility(u3) = eff(u3) \times utility(u2)$$

The utility of a goal unit without successor is fixed to 1.

This method is similar to the one described in [15] or in [13] where dynamic programming is used for planning. In these methods, the systems affects transition probabilities to different couples of states of the environment. Here, the efficiency $eff(u3)$ is also a transition probability, but between two sets of states: $I(u3)$ and $I(u2)$.

6.1.2 adding a unit

Adding a new unit is a more complex process than in the first growing network. Here, several units can have the same successor and the same action. Since the units have fixed weights, the only way to learn the good set of weights is to test several possibilities. The criterion to decide whether a new unit must be generated or not, is less restrictive. A new unit $u4$ can be associated to action a and to the successor $u2$ at time t, if each one of the following conditions is satisfied:

- $u2 = n_{best}(i,t)$ and $Act(u2,t) > 0.9$,
- The action a as just been completed,
- There does not already exist another unit u associated to a, with $u2$ as successor, which satisfies $eff(u) > 0.9$ and $Act(u, t_{decision}) > 0$.

If a unit such as u exists (in figure 2 it would be if $Act(u3, t_{decision}) > 0$ and $eff(u3) > 0.9$), it means that the module has nothing to learn about the current situation, since u was able to predict it. If a unit $u4$ can be created for the successor $u2$ and the action a, its weights will be initialized according to one of these formula:

1. $w_{i,u4} = Min(input_i(t_{decision}), w_{i,u})$, where u is a unit associated to the action a and the successor $u2$, and such that $eff(u) > 0.9$.
2. $w_{i,u4} = input_i(t_{decision})$, if no unit like u is found.

In the first case, the new unit $u4$ is a generalization of a previous efficient unit (for example $u3$ if it is efficient enough). As a matter of fact, the lower are the weights of $u4$, the greater is the set $I(u4)$. (if there is several units satisfying the condition of u, the first found is taken). In the second case, the new unit is a simple memorization of the state where the decision of using a was taken.

7 The experiments

7.1 The task

The task consists in taking a block and putting it in a basket which depends on the digit which is written on it. The system must learn to focus on the block, take it, recognize the digit, decide in which basket to put it, then put it in this basket. The system will have to learn how to manipulate the blocks and how to recognize their digits.

7.2 The agent and its environment

The environment is a 6×7 grid of positions. Each position can contain one basket and/or one block and/or the hand of the agent. There are ten baskets of different colors, one block and one hand. At the beginning of a task, the block is in one of the 8 bottom positions. The baskets are randomly placed in the other positions (figure 3).

The agent can focus on a particular position so that it perceives the location of the different objects relatively to this position. This provides an implicit generalization of the input perceptions. It can move its focus point with four elementary focus-movement actions: one position up, down, left or right.

The agent has one "hand" which can be moved in these four directions with four other actions. The hand can also be opened or closed in order to grab the block if it is at the hand's current position.

Finally, the system has twelve external effectors divided in three groups: the focus movements (up, down, left, right), the hand movements (up, down, left, right), and the grab movements (open, close). The hand and focus movements have a repetition delay of 10 steps.

7.3 The perceptions

The agent perceives its environment through several kind of input detectors. There are location detectors, the retina, the hand detectors and the reward detector.

The location detectors are divided into subsets corresponding to the different objects of the environment: one for the block, one for the hand and one for each basket. A subset of location detectors is composed of nine detectors. One of them detects the object when it is in the focus point. The other ones detect the direction of the object. There is one detector for each one of the following directions: above, below, right, left, and the four diagonal directions above-left, below-left, above-right and

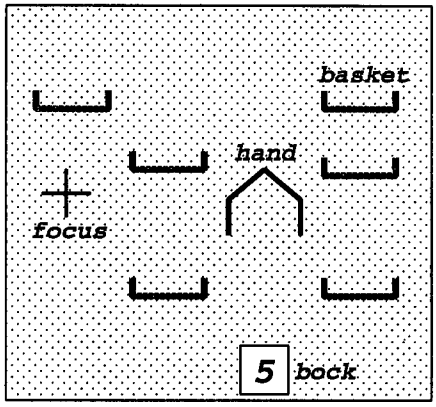


Figure 3: *The agent in its environment. The agent controls the hand and the focus.*

below-right. It is because of the lack of precision of these location detectors that we had to set the repetition delays of the movement actions to 10. As a matter of fact, the focus can move several steps in one direction without producing any modification on the state of the location detectors.

The retina is an input matrix containing the image of the digit when the focus point is on the block. The digits have been framed in a 7×8 grid with 7×8 input coefficients representing the density of ink in the corresponding square of the grid.

The hand detectors indicate whether the block is in the hand or not, and whether the hand is opened or not. The reward-detector is active only when the block is in the good basket (at the same position).

7.4 The modules

Five different modules has been used for learning this task. It is important to determine which kind of input must be related to which kind of output, and at which level. It is also necessary to define the goals of the lower level modules so that what they learn is useful for the higher level modules. It seems evident for example that the localization detectors must be related to the focus movement actions at a low level, in order to learn to focus on different objects. This module must have several goals. Each one consists in focusing on one particular object.

Another important ability consists in moving the hand toward the focus position, in order to learn to bring it on the object which is focused. Then another low level module should realize the association between localization detectors and hand-movement actions. The goal of this module is characterized by the presence of the hand in the focus.

One could think that a single module could be sufficient for these two tasks. But if we want the system to learn how to have the hand in its focus by moving its hand, not by moving its focus, it is then necessary to have a separate module which cannot use the focus-movement actions to achieve this goal.

For the moment, we have not a general way to choose and connect modules in regard to the required task. Our method remains task-dependent. Here is the list of modules that we had to define for this task:

- Mf: Its aim is to focus on an object (basket or block). Its inputs are the location detectors, its actions are the focus movements. It contains 11 goals which consist in focusing on each one of the 11 objects (1 block and 10 baskets).
- Mh: Its aim is to bring the hand in the focus. Its inputs are the location detectors, its actions are the hand movements. Its unique goal consists in focusing on the hand.
- Mb: Its aim is to see both the hand and an object. It controls the modules Mf and Mh. It contains 11 goals which consist in focusing on both the hand and one of the 11 objects.
- Mt: Its aim is to take the block. It controls the module Mb. It also receives the tactile perceptions, and controls the finger actions. It contains one goal which consists in having the block in the hand.
- Mg: Its aim is to see the block in the good basket. It controls the modules Mb and Mt. It also receives the retina inputs and a special reward detector which is activated only if the block is in the good basket. Its unique goal is to receive the reward.

7.5 The experiments

The total corpus of digits contained 560 hand-written digits from three writers. 450 digits were used for learning and the remaining ones were kept for testing.

The learning experiment consisted in presenting an initial state of the environment and waiting until the agent puts the block in a basket. If the block is in the good basket, the reward detector is activated. In any case, once the block is in a basket, a new initial state is presented, with another digit on the block, and the ten baskets in different positions. A new initial state is also presented if the agent fails to put the block in a basket within 1000 time steps (1000 elementary actions).

Each system has been trained in two phases: during the 500000 first time steps, the systems had a rather high probability to explore random actions (at least 10%). Then, during the 400000 following time steps, the random exploration parameter was lowered so that the system did not explore random actions when the activity of the best activated unit was above the threshold of 0.9. It is not possible to estimate the recognition performances during the first phase since there is at least 10% mistakes

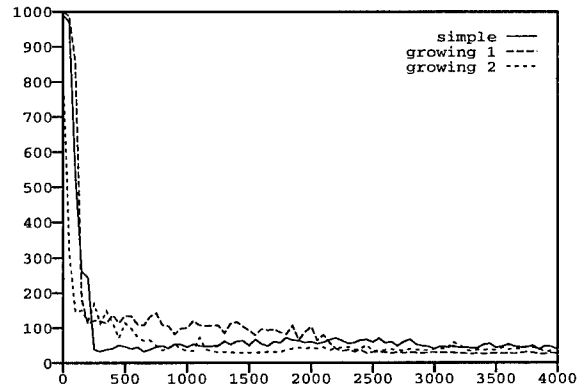


Figure 4: *The comparative number of steps used for each problem. Each point represents the average number of cycles for 50 consecutive problems. The real curve is much more irregular.*

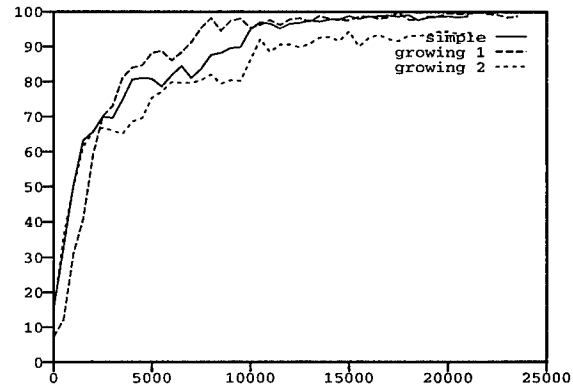


Figure 5: *The comparative results on digit recognition. Each point of the curve has been computed on 500 consecutive trials.*

du to random answers. This is why we wanted to observe the system's behavior when no exploration occurs.

Finally, the test phases were done after the 900000 time steps. During this phase, the training set of digits were replaced by the test set of digits, and the systems were "freezed" so that no learning nor adaptation could occur.

8 The results

The figure 4 shows the comparative number of cycles necessary for each system to bring the block in a basket. At first look the fastest system is the second growing network. But after 2000 problems, the first growing network has its performances stabilized at a better rate. The reason of this result is that the manipulation part of the task does not need the noisy input detectors of the retina to be solved. When the input data have small or little variability, this kind of system is more suitable than a system which learns by slow adaptation of weights.

When the agent starts learning where to put the block, between 250 and 2000 problems (this phase is not decided by the human designer, but normally comes when the agent is able to manipulate the block), the speed of the second growing network is in average similar to speed of the first simple system. In this phase the first growing network is the worst. But this hierarchy changes after 2000 trials, when the agent begins to have a rather good recognition rate. The first growing network stabilizes its performances at a better speed. This remains true until the end of the experiment.

If we look at the recognition rates (figure 5), we can see a similar behavior, the first simple network and the second growing one start faster then the first growing network. This is due to its slow learning speed in the manipulation task. After what, this network needs less examples than the simple one for learning to classify the digits. But at the end, the performances of these systems are similar.

As it was predictable, the second growing network is the worst at digit recognition. Besides, this network is more unstable than the other ones (this does not clearly appear in the figure since the recognition rate is averaged on 500 consecutive trials). The reason of this instability is related to the generate and test process. A unit can give good classification results during a quite long period. Then if bad luck occurs (a sequence of badly recognized digits) the efficiency of the unit drops and it is no more considered by the system for further decisions.

The test results confirm this robustness problem: the digit recognition rate was about 91% for the simple and the first growing network, but it fell down near 60% for the second growing network.

8.1 What is lost compared to supervised learning ?

It is more difficult to learn to recognize patterns through TD learning than through classical supervised learning. To know how much we loose in this experiment, we have compared the recognition rate and learning speed of our systems to a classical single layer perceptron with ten output units similar to the formal neurons used in the first and second systems. The advantage of the supervised learning network is that the ten output units can learn at the same time from the same example. It is not the case in our systems: only the units which has taken the decision can learn from its success or failure. The reason is that nothing indicates to the agent that it faces a pattern classification problem. It cannot consider *a priori* that the digits belong to non overlapping classes. Therefore, we could predict that the number of examples necessary for the supervised network to reach the same performances would be divided by ten.

The perceptron reaches about 98% recognition on the

training set after 400 or 500 examples. It indicates that our TD learning system could not do better than 5000 examples (more if we consider that they cannot learn to classify digits before being able to manipulate the block). But a comparable recognition rate is obtained after only 8000 examples by the first growing network which is the fastest one. It seems to be about 3000 examples too many. Besides, the test performances of the perceptron are about 95%, which is 4% better than our system. More work needs to be done to know if it can be improved or if it is intrinsic to the TD method.

8.2 Conclusion about the results

In the experiment presented here, the fact of learning and using a more explicit representation does not lower the learning speed, despite the greater amount of units that need to be created and tested. On the other hand, the third system which builds a more precise internal representation is clearly less robust to noisy data even if its speed is greater when the input variability is lower. This result was predictable since the units of this system have a discontinuous activation function.

Since the different systems are more or less adapted to different levels of this task, a solution to improve performances could consists in using different kinds of modules within the same agent. This is immediately possible since the three kinds of modules interact in the same way.

It is also possible that the limitations we encountered while developing the second growing network system, come from the philosophy which consists in using the same modules for learning to behave and learning internal representations. Maybe that we could obtain both representation and robustness, if we could use the first kind of modules for learning to react, and another for learning the representation of what the first does.

9 Related works

The use of modular architectures for TD learning has already been explored in different ways: with neural networks by Lin [11] or with classifier systems and genetic algorithms by Dorigo [8]. As we have pointed out during the above sections our method is related to both ones: it uses neural networks, and our second growing network uses a generate and test learning process like genetic algorithms. Millan [6] has also used a growing network technique for real robot learning, but in a very different way, without temporal connections nor modular decomposition.

Our approach has also some similarities with the TRAIL system proposed by Benson [1]. It uses similar tree-like representations with actions that can also have variable durations. The main difference with our approach is that TRAIL is a symbolic system. It can use neural networks for lower level tasks, but as separate modules. Our point of view is that numerical and statistical learning (with neurons or not) is required at any level. For this reason, we prefer to design new kinds of neural networks which support representations and adapt them to high level computation. Our approach is in the field of neuro-symbolic integration [9], but is different from the hybrid approaches which concern architectures containing both symbolic modules and connectionist modules. Our work is rather concentrated on the problem of designing a unified model, based on artificial neurons at any level.

The problem of reusing what has been learned by a neural network in the frame of TD learning has also been studied. S. Thrun [18] proposes to apply Explanation Based Neural Network Learning Algorithm (EBNN) in the context of robot navigation learning. In this system, a first network learns an action model, then a second one uses a Q-learning algorithm to learn how to act. Thrun shows how EBNN can be applied to use the first network to speed up learning in the second one. It shows one interesting way to reuse what is coded in a multilayer perceptron, but it is not evident to propose many other ways. We think that we need more explicit information coding to have more possibilities for further reuse. This is the aim of our second growing network.

10 Discussion

Here are some important questions about the future of this work:

- How could we reuse the representation ?
- How to define appropriate modular decompositions ?
- How will it work with real robots ?

The answer to the first point is partially given in [2] where a representation similar to the one of the third model is used for a perceptually grounded knowledge representation with capabilities similar to symbolic systems.

For the modular decomposition problem, the designer needs to decide which input and output units have to be taken into account by each module. In some cases, finding the good connectivity is easy. For example, the sound perceptions are generally not relevant for object manipulation. But it will not be so easy in any case. Our best chance is that the connectionist TD algorithm can find itself the useful connections. So we think that the research problem is rather to find algorithms which can find a good connectivity, starting from the one given by the designer.

Another problem is the choice of the appropriate subgoals of the module. It seems impossible that in more complex tasks, the designer provides any possible subgoal. Therefore we need some methods to generate subgoals automatically. The automatic subgoal generation is one of the capabilities of symbolic systems such as

problem solvers. This is another argument in favor of learning explicit representations in order to access to the symbolic power within the TD learning frame.

The application to real robots will be more difficult. It is probable that these algorithms will have to be adapted. We hope to have the opportunity to test this system, as is, on a mobile robot soon, but we also want to pursue soft applications (in simulated environments) in order to explore faster different ways to reuse the representations that our system can build. We want to apply it on more complex learning tasks in the cognitive level as well as in the perceptual level. In fact, our approach consists in studying how algorithms similar to the ones used on real robot learning, can be adapted to higher level cognitive process. If our solutions remain close enough to these algorithms, we think that we will have good chances to adapt them to real robots, since we can still use the progresses done on real robot learning [12] [6] [14] [8].

References

[1] Scott Benson. Inductive learning of reactive action models. In Morgan Kaufman Publishers, editor, *Machine Learning: Proceedings of the Twelfth International Conference*, San Francisco, 1995.

[2] P. Blanchet. *Une architecture connexionniste pour l'apprentissage par l'expérience et la représentation des connaissances.* PhD thesis, Université Paris XI, Orsay, France, December 1992.

[3] P. Blanchet. An architecture for representing and learning behavior by trial and error. In *From animals to animats 3, third international conference on simulation of adaptive behavior*, 1994.

[4] Pascal Blanchet and Frédéric Alexandre. A Hierarchical Connexionnist Architecture for Learning Internal Representations of Behaviors. In *Proceedings of the ICANN95 International Conference on Artificial Neural Network*, 1995.

[5] L.B. Booker, D.E. Goldberg, and J.H. Holland. Classifier systems and genetic algorithms. *Artificial Intelligence*, 40(1-3):235–282, September 1989.

[6] José del R. Millan. Learning Efficient Behavioral Sequences from Basic Reflexes in a Goal-Directed Autonomous Robot. In *From animals to animats 3, third international conference on simulation of adaptive behavior*, 1994.

[7] B.L. Digney and M.M. Gupta. A distributed adaptive control system for a quadruped mobile robot. In *From Animals to Animats 3, third International Conference on Simulation of Adaptive Behavior*, 1994.

[8] M. Dorigo. Alecsys and the autonomouse: Learning to control a real robot by distributed classifier systems. *Machine Learning*, 19(3):209–240, June 1995.

[9] Y. Lallement, M. Hilario, and F. Alexandre. Neurosymbolic integration: Cognitive grounds and computational strategies. In M. DeGlas and Z. Pawlak, editors, *World Conference on the Fundamentals of Artificial Intelligence*, Paris, July 1995.

[10] Long-Ji Lin. Self-improving reactive agent based on reinforcement learning, planning and teaching. *Machine Learning*, 8:293–321, 1992.

[11] Long-Ji Lin. Hierarchical learning of robot skills by reinforcement. In *Proceedings of the 1993 IEEE International Conference on Neural Networks*, pages 181–186, 1993.

[12] S. Mahadevan and J. Connell. Scaling reinforcement learning to robotics by exploiting the subsumption architecture. In *Proceedings of the Eighth International Workshop on Machine Learning*, pages 328–332, 1991.

[13] A.W. Moore and C.G. Atkeson. Prioretized sweeping: Reinforcement learning with less data and less time. *Machine Learning*, 13:103–130, 1993.

[14] S. Zrehen P. Gaussier. A Topological Neural Map for On-line Learning: Emergence of obstacle avoidance in a Mobile Robot. In *From animals to animats 3, third international conference on simulation of adaptive behavior*, 1994.

[15] R. S. Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Proceedings of the Seventh International Workshop on Machine Learning*, pages 216–224, 1990.

[16] R.S. Sutton. Learning to predict by temporal differences. *Machine Learning*, 3:9–44, 1988.

[17] G. Tesauro. Practical issues in temporal difference learning. *Machine Learning*, 8:257–277, 1992.

[18] Sebastian Thrun. An approach to learning mobile robot navigation. *Robotics And Autonomous Systems, Special issue on Robot Learning*, 1995.

Learning to Detour & Schema-Based Learning

Fernando J. Corbacho and Michael A. Arbib

Center for Neural Engineering
University of Southern California
Los Angeles, CA 90089, U.S.A.
corbacho@pollux.usc.edu, arbib@pollux.usc.edu

Abstract

In this paper we extend the analysis of learning to detour (Corbacho and Arbib, 1995) by proposing a generalized framework Schema-based learning (SBL) which incorporates general principles of adaptive organization e.g., bootstrap coherence and coherence maximization principles. A schema is an evolutionarily or experience-based constructed recurrent pattern of interaction or expectation (perceptual, motor, reactive, and predictive schemas) with the environment, and coherence is a measure of the congruence between the result of an interaction with the environment and the expectations the agent has for that interaction. SBL attempts to provide a general and formal framework independent of the particularities of implementation, thus allowing the design and analysis of a wide variety of animats. SBL allows the growth of increasingly complex patterns of interaction between the agent and the environment from an initially restricted stock of schemas. SBL also allows for efficient learning by confining statistical estimation to a narrow credit assignment space, so that the system learns in the right ballpark with minimum "scene" statistics while maintaining a high degree of adaptability.

1. Introduction

In this paper we go beyond (Corbacho and Arbib, 1995) by proposing a generalized framework, Schema-based learning (SBL), incorporating general principles of organization. SBL is an attempt to provide with a general and formal framework introducing general definitions of schemas and their related operations which are independent of the particularities of implementation.

The reasons for studying learning to detour in anurans were spelled out in (Corbacho & Arbib, 1995); here let us simply recapitulate the most important ones: (i) SBL is constrained by data on a neuroethologically sound system -as to the task, the environment and the agent, (ii) the work on *Rana Computatrix* - the computational frog, Arbib (1987) - allows for horizontal integration (across many integrated functionalities) and not just vertical integration (action-perception within one central functionality, e.g. prey catching), and (iii) Learning to Detour has proved to be an adaptive process relying on important processes of learning (Corbacho and Arbib, 1995; Corbacho et al. 1996).

Ingle (1976, 1983) and Collett (1982, 1983) have observed that a frog/toad's approach to a prey or avoidance from a threat are also determined by the stationary objects in the animal's surround. A frog or toad, viewing a vertical paling fence barrier (e.g., a row of chopsticks) through which it can see a worm, may either approach directly to snap at the worm, or detour around the barrier. Corbacho and Arbib (1995) (henceforth referred to as C&A) modeled the different behavioral responses to different barrier configurations, as well as the learning involved in the behavioral transitions. Here we sample a few of our observations of the main capabilities of frogs for detour behavior which set challenges for our learning model. We refer the reader to (Corbacho et al., 1996) for more details.

Experiment I: Barrier 10 cm Wide

Observation 1: Frogs that started from a long enough distance (15-25 cm) in front of a 10 cm wide barrier (and with the worm 10 cm behind the barrier) showed (in 95% of the trials) reliable detour behaviors from the first interaction with the 10 cm barrier. That is, they produced an immediate approach movement towards one of the edges of the barrier (see Fig. 1A).

Experiment II: Barrier 20 cm wide

From now on we will refer to a frog which has not been exposed to the barrier paradigm as naive.

Observation 2: If the chopsticks are placed the same distance apart, so that the gaps have the same width, and the barrier is 20 cm wide, then the naive frog tends to go for the gap in the direction of the prey (this was the case for 88% of the trials). The frog starts out approaching the fence trying to make its way through the gaps. During the first trials with the 20 cm barrier the frog goes straight towards the prey thus bumping into the barrier. When the frog is not able to go through a gap towards the prey it backs-up about 2 cm and then reorients towards one of the neighboring gaps with no apparent bias (see Fig. 1B).

Observation 3: After 2 (43%) or 3 (57%) trials, the frog is already detouring around the barrier without bumping into the barrier (see Fig. 1C). The behavior involves a synergy of both forward and lateral body (sidestep) movements in a very smooth and continuous single movement.

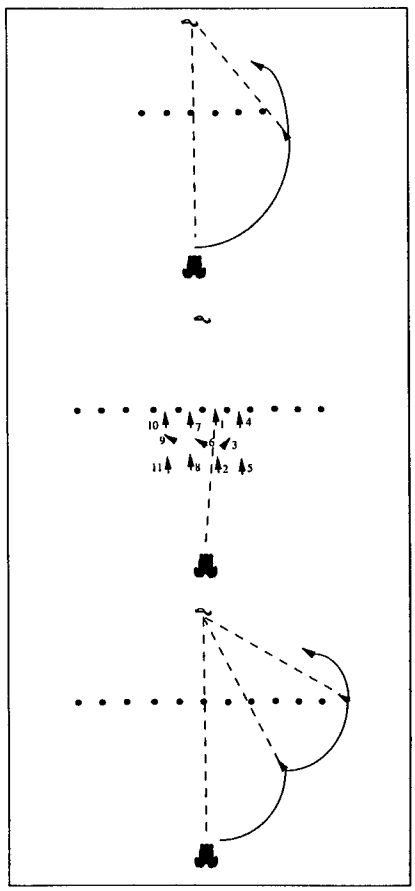


Figure 1. A. Approach to prey with single 10 cm barrier interposed. **B.** Approach to prey with single 20 cm barrier interposed: first trial with frog in front of 20 cm barrier (numbers indicate the succession of the movements). **C.** Approach to prey with single 20 cm barrier interposed: after 3 trials with frog in front of 20 cm barrier. Arrowheads indicate the position and orientation of the frog following a single continuous movement after which the frog pauses. This figures show actual "typical" trajectories from start to finish (traced from video).

2. Schema-based Model for Learning to Detour

Animal behavior is controlled by significant patterns of interaction which are useful again and again, e.g., grasping. Schemas may be "wired-in", or may start as approximations to what have been the relevant patterns significant for the agent in its past interactions with the environment. They provide a somehow vague description of a situation and action, and on the other hand are specific enough to be applicable. They represent what is stable and therefore generalizable over variability, becoming precise through adaptation. We propose that the design of the overall agent (animal or animat) should be done (has been done by evolution) by aggregation of both units of interaction with the environment, as well as units of expectation about those interactions (formed from successful experiences in similar interactions).

2.1 Varieties of Schemas

Perceptual Schemas

Perceptual schemas recognize a context of interaction. Examples of perceptual schemas include the Stationary Object recognition (SOR) and Prey recognition schemas (PREY). For instance the presence of prey within the visual field of the agent produces a 2D pattern of activity in the PREY schema encoding the position and type of the prey (an instance of this schema), while absence of prey leaves this schema at rest. Other perceptual schemas include LOOMING and TACTILE (where TACTILE = 1 if something is touching the agent and TACTILE = 0 otherwise). They include the corresponding mappings that discriminate the different subspaces within the high-dimensional views of the environment (e.g., dilating patterns, and so on).

Motor Schemas

Motor schemas are functional units capable of changing the agent and environmental parameters. C&A introduced the FORWARD motor schema which when active produces a movement of the agent in the direction of the midsaggital axis of the body with frontal direction. SIDESTEP produces a movement orthogonal to the sagittal midline. BACKUP is similar to forward but in the opposite direction. ORIENT changes the agent's field of view by changing the body orientation while maintaining the other coordinates.

Reactive Schemas

Other types of schemas deal with spaces intermediate between perceptual and motor. These spaces having in many cases the "role" of integrating and/or coordinating signals coming from a variety of spaces. C&A describe reactive schemas such as OBSTACLE-AVOIDANCE or PREY-APPROACH linking the perceptual and the motor frames of reference. For instance OBSTACLE-AVOIDANCE relates a certain space of percepts (namely obstacles) with an intermediate (sensory-motor) space. So that information about obstacles can be integrated/ coordinated with other information (e.g. prey information) to converge onto one target motor set. Cobas and Arbib (1992) postulated that the motor heading map (MHM) provides a basis for integrating signals from multiple sensory circuits (see also Liaw and Arbib (1993). In our system, PREY, which encodes prey position, projects excitatory signals to indicate approach behavior (prey-attractant-field), and SOR projects inhibitory signals for avoidance behaviors (SOR-repellent-field).

This set of perceptual, motor and reactive schemas can explain basic facts about detour behavior (C&A): If the retinotopic representation of the edge of the barrier in SOR falls within the prey-attractant-field spread, then (i) the summation of activity from the prey-attractant-field

and from the SOR-repellent map on MHM at the retinotopic position just beyond the barrier's edge is stronger than (ii) the summation at the "center" of the barrier where the prey is located. Hence, the winner-take-all dynamics will select the cluster of activity corresponding to the retinotopic position of the edge of the barrier, thus predicting that frogs would detour around narrow barriers (see Fig. 2).

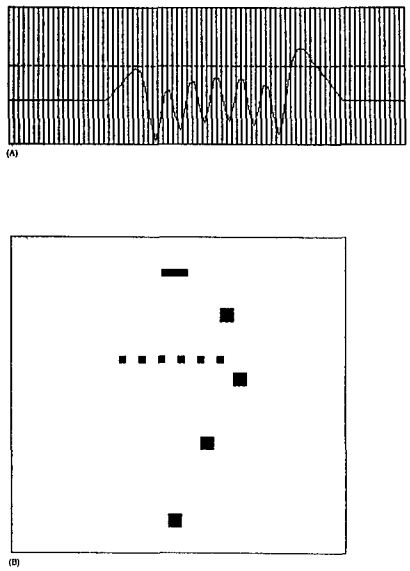


Figure 2. General pattern for approach to prey with single barrier interposed. **A.** Target Heading angle representation in MHM after winner-take-all dynamics settle down when the agent is approaching the 10 cm barrier. **B.** Overall behavior (compare results in Fig. 1A). Notice how the animal "avoids" the barrier to approach the worm.

On the other hand, for wide barriers the prey-attractant-field extent falls within a much wider barrier field. Hence, at the MHM retinotopic position corresponding to the barrier's edge there will be no input activity from the prey map. On the other hand, there will be a great projection of activity on MHM at the retinotopic position of the prey; and this in turn will trigger approach to a point within the barrier map, so long as the peak of prey attraction exceeds the trough of barrier inhibition (see Fig. 3). Thus, the model predicts that the naive frog would approach wide barriers rather than detour around them.

Predictive Schemas

Perception involves a continual updating of our initial comprehension of the more salient aspects of the current environment/situation by noting discrepancies between what we expect and what our senses now tell us. A schema provides abilities for recognition and guides to action but must also provide expectations about what will happen so that we may choose our actions appropriately (Arbib, 1992). These expectations may be wrong, and so it is that we sometimes learn from our own mistakes. We now define a *predictive schema* as a unit of perception, action and adaptation which relates percepts and actions in an adaptive manner, not mere units of interaction but also of expectation. The structure of a predictive schema is similar to that described by Becker (1973). Predictive schemas have a context, an action and an expected/predicted result.

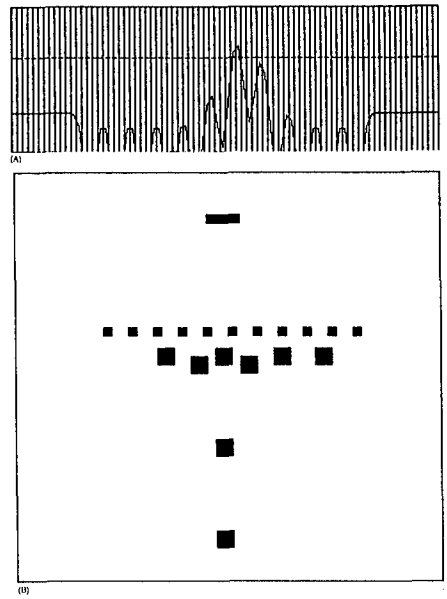


Figure 3. A. First trial approaching to prey with single 20 cm barrier interposed. Target Heading angle representation in MHM. **B.** Overall behavior (compare to results in Fig. 1B). Notice that the animal aims for a (too narrow) gap in the fence in the direction of the prey.

2.2. Efficient learning in SBL

We now extend the previous study on reactive navigation to analyze Behavioral observation 3 on learning that occurs when the agent is presented with a wide barrier in the pathway to its goal. In doing this, SBL will try to address two of the most fundamental problems in adaptive systems, namely, how to decide what substructure(s) is (are) to blame when the system is in error, and how to fix it (them) (cf. credit assignment).

It is advantageous (efficient) to cope with new experiences based on positive results from previously similar experiences. It is efficient since the agent avoids the statistical estimation required to form again past successful action-perception correlations. The current stock of schemas allows the performance of a broad-brush analysis of the action-perception "scene" e.g., prey detection/catching, crude grasping in babies, etc. SBL does not replace statistical estimation, rather, it confines it to a narrow search space, so that the system learns in the right ballpark with minimum scene statistics.

When the agent interacts with the environment, many of its component structures will be undergoing major changes in their patterns of activity. We claim that

a system initially designed to actually expect most of these changes (e.g. shift of activity in Superior Colliculus before movement completion) will have a much reduced space of unexpected changes where the adaptation can be "played", thus, reducing the space for credit assignment. The space of potential "culprits" is mainly reduced to the space of incoherent units - and, hence, will require minimum statistical estimation - where *coherence* is a measure of congruence between the results of an interaction with the environment and the expectations the agent has for that interaction.

2.3. Bootstrap Coherence Principle

Bootstrap coherence consists of providing the agent with the ability to anticipate the results of its basic actions and, thus, expect certain changes in its internal representations accordingly. Initially the agent comes equipped not only with units of interaction but also with *dual* units to predict the result of the corresponding interaction. We have introduced the corresponding predictive schemas linked to associated perceptual, motor or reactive schemas to allow the agent to begin its interactions with bootstrap coherence. Thus, SBL needs to "represent" both the predictive response and the actual observed responses. For a space Q we introduce $\hat{Q}$ as the space of expectations about Q. $\hat{Q}$ can be thought as a "working memory" of Q. A particular instance e.g., $\hat{q}(t)$ represents the expectation for $q(t+1)$. This allows for the comparison/match of both representations, hence, detecting incoherences that will be the trigger for adaptation (coherence maximization principle in section 5).

A note on notation. SOR-MHM represents the obstacle avoidance reactive schema (SOR sends inhibitory activity to MHM). We will also later represent it as a mapping

$$\text{OBSTACLE-AVOIDANCE}: SOR(t) \rightarrow MHM(t)$$

SOR-MHM-TACTILE represents a predictive schema related to the reactive schema SOR-MHM with expectation space for TACTILE.

$$SOR \times MHM \rightarrow \hat{TACTILE}$$

So the predictive schemas are the *dual* units associated with the different perceptual, motor and reactive schemas. To "start" our agent with bootstrap coherence we have included the following *dual* predictive schemas:

i) For every[1] combination of perceptual and motor schemas we have included a *dual* predictive schema to reflect upon the changes that a given motor action exerts over the state of the particular perceptual schema.

$$P(t) \times M(t) \rightarrow \hat{P}(t)$$

where $P(t)$ is the state of a perceptual schema at time t and $M(t)$ is the state of a motor schema at time t. For instance

$$PREY - M: PREY \times M \rightarrow \hat{PREY}$$

where M is a motor action and $\hat{PREY}$ anticipates the pattern in $PREY$ for that action (analogously for SOR-M : $SOR \times M \rightarrow \hat{SOR}$).

ii) Several predictive schemas have been included to take into consideration what the system expects when activating the corresponding reactive schema.

$$P(t) \times I(t) \rightarrow \hat{P}'(t)$$

Notice that P' is not necessarily the same as P. The predictive schema "checks" whether the reactive schema performs as expected -is the result of the interaction with the environment as expected?

As an example, let us describe the SOR-MHM-TACTILE predictive schema. If the OBSTACLE-AVOIDANCE schema works appropriately, then the agent should not detect tactile stimulation due to bumping into the stationary objects perceived, that is, SOR-MHM-TACTILE will be coherent. Notice the difference with simple error-detecting systems. In SBL not every tactile stimulation corresponds to an incoherence (error) but only certain tactile stimulation under certain conditions corresponds to an incoherence. In any other circumstances the same tactile stimulation would not be considered an error (e.g. it may actually be expected).

At any point in time the system may have many schemas active yet many of these schemas are in a coherence state, thus, the system notices the few incoherences (unexpected changes) easily. Giving bootstrap coherence and "continuous" environments, the space of unexpected changes should be smaller than the space of changes.

2.4. Seed Schemas for Learning to Detour

We introduce several new schemas beyond C&A, namely LOOMING-RECOGNITION (Liaw & Arbib, 1993), LOOMING-AVOIDANCE, LOOMING-AVOIDANCE-TACTILE, & PREY-MHM-TRIGEMINAL (prey catching predicts prey in mouth); as well as all the predictive schemas needed to allow for bootstrap coherence. In summary the agent is designed with the following *seed* schemas:

Perceptual: {SOR, PREY, TACTILE, LOOMING}

Motor: {FORWARD, SIDESTEP, ORIENT, BACKUP}

Reactive: {PREY-APPROACH,OBSTACLE-AVOIDANCE, BUMPING-AVOIDANCE, LOOMING-AVOIDANCE}

[1] In more complex agents only a subset of the overall set of possible combinations might be included (Corbacho & Arbib, 1996).

Predictive: {PREY-MHM-TRIGEMINAL, PREY-M, SOR-MHM-TACTILE, SOR-M, LOOMING-MHM-TACTILE, LOOMING-M}

Several perceptual schemas may be active sending signals to MHM, after dynamics settle down a target heading angle is produced, making it unclear which perceptual schema is responsible if that target heading angle gives raise to some incoherence - there is a need to keep composites separate. Also several actions might be performed within the same "time window" e.g. sidestep, backup and orient, yet only a subset of actions is responsible for the incoherence. Hence, the different subconfigurations must be kept separate.

We introduce an input *composite* by introducing several inputs simultaneously giving raise to the *superposition catastrophe* (von der Malsburg, 1986) where different representations are undifferentially merged. One of our proposed solutions is through the use of labels for the different components of the composite which originate from different functional units. In physiological terms this might correspond to different temporal structure on the incoming signals representing the different components.

There often exists the possibility of constructing/tuning too many different combinations of schemas unless the incoherence can be traced back to the "culprit" subconfiguration. The "synchrony code" enhances a particular subconfiguration to make it available for one-shot learning. In this way a whole configuration can be stored as all the component schemas have the same label. For instance when TACTILE=1, two schemas are in incoherence, namely SOR-MHM-TACTILE and LOOMING-MHM-TACTILE, which one should be tuned? In some cases both, but in some other cases only one of them e.g. looming pattern in the periphery and obstacle (SOR) in front of (touching) the agent. In this last case it is the SOR-MHM reactive schema that should get tuned. The different labels serve to keep the different configurations "separated" so that adaptation of the proper ones can be efficiently performed (e.g., relaxation labeling in section 5.1).

3. Transitions and Learning Sequence

The present section provides the most relevant results of the simulations designed to replicate Behavioral Observation 3 on learning that occurs when a naive frog is repeatedly presented with a wide (20 cm) barrier separating it from a worm. The adaptation of the system is reflected as a sequence of schema tuning and/or construction when incoherences arise. Schema tuning corresponds more to "parametric" changes within a schema (e.g., skill refinement) rather than to any structural change or creation of a new unit of interaction (e.g., skill formation) which corresponds more to schema construction (refer to section 5 for a complete formal definition). Given a current interaction with the environment, the dynamics of the system "measures" the coherence of the expectation with respect to the current results of the interaction, and tries to maximize the coherence by adaptation.

The view field of the agent after having approached the barrier (Fig. 4A) contains a gap which triggers an approach (forward) action towards it (Fig. 4B). Upon reaching the gap, the agent bumps into the fence, thus, making the TACTILE schema active.

Incoherence 1

Since both TACTILE and SOR-MHM are active the predictive schema SOR-MHM-TACTILE reaches an incoherence:

$$TACTILE(t+1) = 1 \text{ yet } \hat{TACTILE}(t) = 0$$

$$\xi_k(t+1) = \left| TACTILE(t+1) - \hat{TACTILE}(t) \right|$$

$$\xi_k(t+1) > th^k \text{ where } th^k \text{ is some threshold}$$

That is, the incoherence is larger than a certain threshold (refer to section 5.2 for a complete formal definition). As a result the reactive schema SOR-MHM gets tuned. C&A discuss in detail the tuning of this reactive schema by tuning its schema mapping -implemented by a connectivity kernel that becomes more inhibitory.

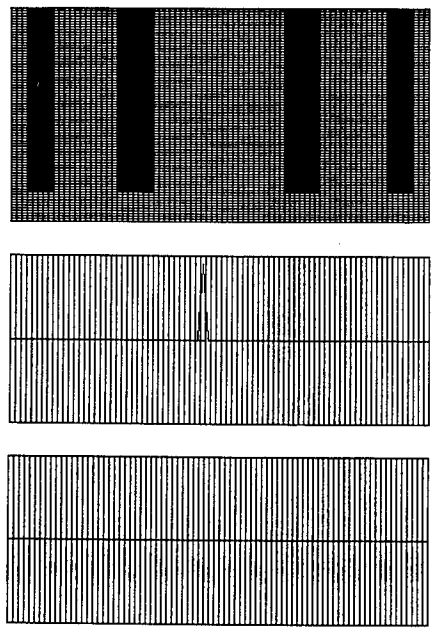


Figure 4. A. The view when the agent is close to the barrier. **B.** The activity in MHM before adaptation reflecting the target heading angle for the gap in front of the agent. **C**. Activity in MHM after adaptation reflecting the "impasse", i.e., no target heading angle. The central gap appears wider since the perceived gaps width decreases with eccentricity.

After the increase in inhibition the system reaches an *impasse* in MHM (Fig. 4C). Namely the system has learned that these gaps should not be approached. Then,

the system dynamics give rise to "exploratory" behavior (biased by perceptual gating; see C&A for details) which as a result eventually produces an action to be taken (in our example FORWARD+SIDESTEP motor synergy). After taking several sidesteps the agent happens to be close to the end of the barrier (Fig. 5A). At this point the same strategy of FORWARD+SIDESTEP unexpectedly gets the agent at the end of the barrier. This gives rise to an *incoherence* as the expected result (the impasse activity in Fig. 4C) of the action does not match the observed result (target heading angle in Fig. 5B). As a result of the incoherence the system triggers the construction of a new predictive schema.

Incoherence 2

$$\xi_k(t+1) = \left| MHM(t+1) - \hat{MHM}(t) \right| > th^k$$

The unexpected change triggers the construction of a new predictive schema consisting of

a *context* $q^i(t+1) = MHM(t)$;

an *action* $q^j(t+1) = SIDESTEP(t) \& FORWARD(t)$

and an *expected result* $\hat{q}^k(t+1) = MHM(t+1)$.

This new predictive schema then starts as a gross approximation of the interaction between the agent and the barrier and eventually, through tuning, "converges" to a more coherent depiction of the interaction.

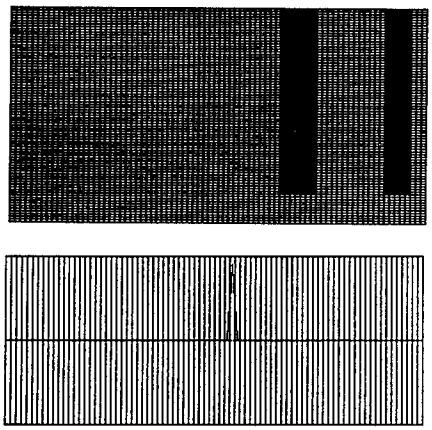


Figure 5. A. The view when the agent is close to the barrier and on the edge. **B**. The activity in MHM reflecting the large gap to the side of the barrier.

Incoherence 3

When the agent is placed in the initial position (20 cm from the barrier) the new detour predictive schema gets activated, thus taking the corresponding action, namely the forward-sidestep synergy. The result is that the frog ends up closer to the barrier in an eccentricity position intermediate between the prey and the barrier edge. SIDESTEP may not take the agent all the way to the edge of the barrier (the amplitude is not large enough). Thus, an incoherence due to an incorrect expectation arises, namely the cluster of activity is not centered in MHM since the animal is still placed lateral to the edge. The predicted result and the observed result do not exactly match. The observed result is actually a *shift* of the predicted result. This incoherence triggers *tuning* of the detour predictive schema. The sidestep amplitude in the detour predictive schema is then revised proportional to the error (C&A). Thus, in the next activation of the detour schema, the sidestep component will have a larger amplitude.

4. Schema-based Learning Architecture

We now go beyond the Learning to Detour case study to begin to provide the more general SBL framework. In this short paper we will only be able to sketch the formal SBL framework. We refer the reader to Corbacho and Arbib (1996a) for a more detailed exposition. We claim that any agent must be analyzed (designed) with respect to its environment. The environment provides the agent with an interaction space. Ultimately the behavior of any agent can only be understood in relation to the synergy agent-environment.

Definition. An *Environment* E is a space which includes a collection of entities and their relations (interactions). A particular instance configuration of E at time t will be denoted as $e(t)$.

C&A introduced a specific E namely, a 150x150 grid where different entities (e.g., barrier, frog) interact. The simulation system contains simplified *environment* functions designed to allow for an adequate interaction between the simulated agent and its environment, for instance the simulation system performs simple "shifts" of the agent's visual field as it moves in the environment and its coordinates change. Part of the environment function is the mapping $F: E \rightarrow V$, where $V \subset \Re^{n \times n}$ is the n^2 dimensional space of views from that environment. A particular instance $v(t) = F(e(t))$ corresponds to a particular "view" of $e(t)$ such that $v(t) \in \Re^{n \times n}$ is a point within the space of views V. The input space is continually cycled through. Basically, the visual field of the agent is determined by a sector of $e(t)$, and the coordinates of this sector are updated as the agent moves around.

Definition. An *Adaptive Autonomous Agent* (*AAA*) is a four-tuple $(P(t), M(t), I(t), S(t))$ where $P(t)$ is the Perceptual space, $M(t)$ motor space, $I(t)$ intermediate space, and $S(t)$ is a collection of different seed schemas relating the different spaces (defined below).

Each space is also composed of subspaces, for instance the perceptual space consists of several (sub)spaces $P=\{P^1,P^2,...,P^l\}$[2] for finite l and $P^i \subset \Re^{m_i \times m_i}$ $(m_i < n)$ dimensionality reduction. A perceptual (sub)space will be defined by a particular set of features (e.g. size, motion). The spaces we deal with in Brain Theory are mainly 2D maps, i.e. neural layers. The actual pattern of activity in the layer is an instance -element in the space. Different mappings allow one to go from one space to another.

Definition. A *schema* $S^i=(L^i,\Pi^i,th^i)$ is determined by the schema layer L^i (the instance space, typically a 2D map), and a mapping Π^i which relates the schema layer with other schema layers. Also each schema has an associated *threshold* th^i to determine when the schema is instantiated (schema instantiation will be further discussed in Section 5.3).

Definition. A *schema instance* $s^i(t)$ consists of a particular pattern of activity in the schema layer $L^i(t)$ containing the corresponding instance parameters of the particular interaction (e.g. size, location). The instance has also an associated activation variable A^i reflecting its activation level. Each instance must have its own *activation variable* label since different instances of the same schema may be active at the same time and be part of different schema assemblages. The activation variable can take different activation levels regardless of the parameters of the instance in the schema layer (quasi-orthogonal representations).

4.1. Schema Types

Definition A *perceptual schema* S^i is determined by $L^i \in P$ and $\Pi^i_P: V \rightarrow P^i$, from a particular view $v(t)$ the schema mapping gives raise to the different instances $p^i(t)=\Pi^i(v(t))$ where $p^i(t) \in \Re^{m_i \times m_i}$ $(m_i < n)$ contains the parameters corresponding to that particular instance by the pattern in the 2D schema layer.

Definition. A *motor schema* S^i is determined by $L^i \in M$ and $\Pi^i_M: M^i \rightarrow E$. In general $\Pi^i(m^i(t))=\Delta\overline{\beta}^i$ where $m^i(t)$ corresponds to an active motor schema instance and $\overline{\beta^i}$ is a vector of parameters in E related to $m^i(t)$.

[2] From now on we drop the t (time) to simplify notation. We are dealing with adaptive systems so we allow all their components to change in time.

Definition. A *reactive schema* S^i is determined by $L^i=Q^l$ where $Q^l \in I \cup M$, and $\{Q^\alpha\}_{\alpha \in J}$ is an indexed family of spaces in $P \cup I$.

Definition. A *predictive schema* S^i is determined by $L^i=\hat{Q}^k$, $\Pi^i: Q^i \times Q^j \rightarrow \hat{Q}^k$ where $Q^i, Q^k \in P \cup I$; $Q^j \in I \cup M$, and $\Pi^i \in \Pi_E$.

For instance the Detour Predictive schema consists of

$$q^i(t+1)=MHM(t)$$

$$q^j(t+1)=FORWARD(t) \oplus SIDESTEP(t)$$

$$q^{\hat{k}}(t+1)=M\hat{H}M(t)$$

The Domain of the predictive schema mapping indicates which (perceptual, motor, and/or reactive) schemas it is associated with.

5. Schema Functions

5.1. Schema Dynamics

By introducing dynamic activation variables for each schema instance we make the system subject to self-organization, able to relax into different structured states (assemblages) that represent structured higher-level schemas. The activation variables allow for processes of self-organization to occur in the dynamics time scale (von der Malsburg, 1981), thus forming different assemblages within the overall topology in the fast time scale. Since different assemblages might coexist in the dynamics scale, and yet be differentiated; there must be a way to express relations between active schemas which belong to the same assemblage (as for instance representing parts of the same concept), thus avoiding the *superposition catastrophe* (von der Malsburg, 1986) where different assemblies are undifferentially merged. The dynamics of SBL follow the Relaxation Labeling framework (Hummel & Zucker, 1983) where the nodes are the schema instances and the labels assigned to each node represent the different *temporal signatures* assigned to the activation variable. Relaxation labeling then allows for the instantiation and de-instantiation of schemas as well as dynamic formation of assemblages of schemas with common temporal signature.

5.2. Relaxation Labeling

Relaxation Labeling assigns labels to objects. They are processes that reduce ambiguity and noise, and select the best labeling among several possible choices. In a labeling problem, one is given: (1) a set of objects; (2) a set of labels for each object; (3) a neighbor relation over the objects; and (4) a constraint relation over labels at

pairs (or n-tuples) of neighboring objects. Generally speaking, a solution to a labeling problem is an assignment of labels to each object in a manner which is consistent with respect to (4). We will define the schema instances activation variables as the objects $A=\{A^1,A^2,...,A^C\}$, a label set $L=\{l_1,l_2,...,l_L\}$ associated with each activation variable, and a set of *compatibility* functions $R_{i,j}:L\times L\rightarrow\Re$ where $R_{i,j}(l,l')$ is defined as the relative *support* (either positive or negative) that label l' for activation variable j offers for label l for activation variable i. A labeling assignment is a map $B:A\times L\rightarrow\Re$ where $B_i(l)$ is the confidence that activation variable i should be labeled with l. We further require that $0\le B_i(l)\le 1$ for all i,l and that $\sum_{\alpha}B_i(l_\alpha)=1$, yet they are not to be interpreted as probabilities. The support for label l at activation variable i by the assignment B is defined by

$$Q_i(l)=Q_i(l,B)=\sum_{j=1}^{C}\sum_{l'=l_1}^{L}R_{i,j}(l,l')B_j(l')$$

To update B we then use a simplified algorithm introduced by Rosenfeld, Hummel and Zucker (1976), the new assignment values are defined to replace the current values according to

$$B_i(l)\Leftarrow\frac{B_i(l)[1+Q_i(l)]}{\sum_{l=l_1}^{L}B_i(l)[1+Q_i(l)]}$$

The schema activation variable A^i may have any of the values in L. Their associated $B_i(l)$ indicate the confidence on that particular label, and the different labels correspond to different temporal signatures -i.e., different temporal structures. That is, an activation variable is not just more or less active but the temporal pattern of activity may be different regardless of the activation level. In particular we have defined labels which correspond to inactive l_0, gated l_1, and different temporal signatures l_j $j>1$. We assume that different temporal signatures arise in the perceptual system and these are propagated so that elements with the same origin will have similar temporal signatures, and vice versa.

5.3. Coherence Maximization

A *predictive schema* $S_E^k=\left(\hat{Q}^k,\Pi^{\hat{k}},th^k\right)$ is in incoherence iff $\xi^k(t+1)=\left\|q^k(t+1)-\hat{q}^k(t)\right\|>th^k$ where $q^k(t+1)\in Q^k$, $\hat{q}^k(t)\in\hat{Q}^k$, in the case $\Pi^{\hat{k}}:Q^i\times Q^j\rightarrow\hat{Q}^k$ is the predictive schema *dual* to the reactive schema $\Pi^k:Q^h\rightarrow Q^k$ then

$$\xi^k(t+1)=\left\|\Pi^k(q^h(t+1))-\Pi^{\hat{k}}(q^i(t),q^j(t))\right\|$$

In other cases the predictive schema may be the *dual* of a combination of perceptual, motor and/or reactive schemas. We refer the reader to (Corbacho & Arbib, 1996) for more details. The dynamics of the system tend to minimize incoherences ξ^k by *adaptation* by the different SBL Schema Operations described below.

5.4. SBL Schema Operations

Definition. A schema $S^i=\left(L^i,A^i,th^i\right)$ is *instantiated* when $A^i(l_j)>th^i$ and $j>1$, where $A^i(l_j)$ depends on the *support* from the other related schema instances. The instance will then correspond to the pattern in the schema layer L^i i.e. $l^i(t)$. The parameters of the instance are coded in the specific spatio-temporal pattern. For instance the motor schemas action on the controlled "musculature" is only enabled when their *activation variable* surpasses the threshold.

Definition. A schema consisting of a mapping S_F is *tuned* iff $S_F(t+1)=\Phi\left(S_F(t)\right)$ where Φ is a homototopy (cf. Munkres, 1975).

Definition. Schema *construction* builds a new mapping $\Pi^i\in\Pi_E$ $\Pi^i:D\rightarrow R$ with Domain $D=Q^i\times Q^j$ and Range $R=Q^{\hat{k}}$, thus, defining a new relation among 3 spaces.

The system can construct predictive schemas given a current context state and the current observed result, so that the current observed result becomes the predicted result in future applications of the schema in the same context. To store the results in the past allows to build expectations in the future.

We have used an associative memory (Hertz et al., 1991) to implement the mapping relation for the newly constructed predictive schema. The associative memory stores the predictive schema component elements $q^i(t),q^j(t),q^{\hat{k}}(t)$ as the component vectors of the input pattern ξ^k. To get the system to recall the most similar of several patterns we make $w_{i,j}$ a superposition of terms, one for each input pattern:

$$w_{i,j}=\frac{1}{N}\sum_{\mu=1}^{T}\xi_i^\mu\xi_j^\mu$$

where T is the total number of stored patterns labeled by μ. When retrieving past experiences the input to the associative memory may consist of any combination of component input vectors of the overall input pattern, e.g. the context only, the prediction only, and so on, The rest of the pattern will be completed, thus, retrieving the missing components of the predictive schema.

6. Conclusion

That is, the system ends up with its Obstacle Avoidance schema tuned to better discriminate between passable and not passable gaps. The system also ends up with a new predictive schema which allows the system to be able to predict how to get out of *impasse* in its motor heading map by taking a sidestep/forward motor synergy. The model generates several predictions:

Prediction (behavioral). Upon removal of one of the lateral fence-sticks in a 30 cm wide barrier the agent goes about 50% of the times towards the side where the stick is removed and 50% of the times to the other side. This is the result of a sidestep triggered by the predictive detour schema and, thus, not based on specific perceptual cues about the edges of the wide barrier. This prediction has already been tested by Corbacho et al. (1996).

Prediction (physiological). Some cells in MHM (tegmentum) show an increase in firing activity anticipating the appearance of the barriers edge in response to an intended sidestep movement. We refer the reader to (Duhamel et al., 1992) for anticipatory responses to intended eye movements in LIP neurons.

A general question is how the system "decides" which structures are modulated, and what type of modulation is exerted upon the "chosen" structures. In general the "culprit" may come from different structures, and thus the system faces a *structural credit assignment problem* - the problem of deciding which subset of those structures should be adapted and how. We have shown how to restrict the space for Credit Assignment, and, thus, Reduce Statistical Estimation. Only Incoherent Units define initial Credit Assignment Space. The result is a "almost one-shot" (2-3 trials) learning, system dynamics focus onto the right structures. We also claim that as long as the Bootstrap Coherence is maintained the system scales up without introducing credit assignment problems.

The model presented does not include much in terms of *evaluation*. Future work should extend the formulation so that the schema chosen trades off closeness of context with some measure of success of the invocation of the schema action in that context.

6.1. Comparison to related approaches

Learning to detour could be implemented in a backpropagation network (Rumelhart et al., 1986), where the input map would be the same as the current input to our perceptual maps, and the output map would be the motor action the system is to perform. The problem, then, becomes clear: the learning time for the backpropagation network to "converge" to some reasonable solution would be very large; and it would grow exponentially for even larger systems (bad scaling up). This is mainly due to the lack of structure in the weight space of the backpropagation network. On the other hand, our system learns in one to three trials as the learning space is very focused by the structure (schemas). The credit assignment space has been reduced to the space of unexpected changes.

For related approaches to learning in Artificial Intelligence (AI), refer to Carbonell & Gil (1990), Drescher (1991), and Shen (1994). All of them were greatly influenced by Piaget's work on development in the child (Piaget, 1954). They all provide autonomous learning from the environment, as we claim in this paper, as well as using expectations to "help" adapting the system. On the other hand, they suffer the "brittleness" of AI symbolic representations. As we move towards more biologically plausible systems, more robust and efficient storing (Sokolov, 1975) and matching (von der Malsburg, 1988) mechanisms will be necessary.

The reactive navigation component here described is similar to the potential field method employed on the mobile robotics circle. For instance Arkin (1989) AuRA's path planning model is based on obstacles exerting repulsive forces onto the robot, while targets apply attractive forces to the robot. Our work on the design of autonomous agents also has several points in common with the autonomous robots community. For instance we share with Brooks (1986) the behavioral decomposition and the absence of a single central representation. This is achieved in SBL by the schema decomposition and the assemblage of schema instances. We also attempt to provide both adaptivity and dynamics within a formal framework beyond that of Maes and Brooks (1990). We claim that SBL supports more efficient learning and allows for more complex dynamic construction of behaviors.

Acknowledgments

Preparation of this article was supported in part by award no. IBN-9411503 for Collaborative Research (M. A. Arbib and A. Weerasuriya, Co-Principal Investigators) from the National Science Foundation.

References

Arbib, M. A. (1987). Levels of modeling of visually guided behavior. *Behav. Brain Sci.* 10, 407-465.

Arbib, M. A. (1992). Schema Theory. In: The Encyclopedia of Artificial Intelligence. (Shapiro, S., ed.) 2nd Edn pp. 1427-1443. New York, NY: Wiley Interscience.

Arkin, R. (1989). Motor Schema based mobile robot navigation. *Int. J. Robotics. Res.* 8(4), 92-112.

Becker, J. (1973). A model for the Encoding of Experiential Information. *Computers Models of Thought and Language*, eds. Schank, R. and Colby, K. pp.396-434. San Francisco: Freeman.

Brooks, R. A. (1986). A robust layered control system for a mobile robot. *IEEE Trans. Rob. Automation*, 2: 14-23.

Carbonell, J. G. & Gil, Y. (1990). Learning by experimentation: The Operator Refinement Method. (In *Machine Learning: An Artificial Intelligence Approach*. Vol. III. Y. Kodratoff & Ryszard Michalski Eds.). Morgan Kaufmann: San Mateo.

Cobas, A. & Arbib, M. A. (1992). Prey-catching and predator-avoidance in frog and toad: Defining the schemas. *J. Theor. Biol.*, 157, 271-304.

Collett, T. (1982). Do toads plan routes? A study of detour behavior of *B. viridis*. *J. Comp. Physiol. A*, 146:261-271.

Collett, T. (1983). Picking a route; Do toads follow rules or make plans? *(Advances in Vertebrate Neuroethology,* J.P. Ewert, R.R. Capranica and D.J. Ingle, Eds), pp.321 - 330.

Corbacho, F. & Arbib, M. A. (1995). Learning to Detour. *Adaptive Behavior*, 3(4), 419-468.

Corbacho, F. & Arbib, M. A. (1996). Schema-based Learning. (in preparation).

Corbacho, F. J., Khort, B, Nothis, A. and Arbib, M. A. (1996). *Learning to Detour: Behavioral experiments with frogs*. Technical Report, Center for Neural Engineering, University of Southern California, Los Angeles, CA (in preparation).

Cunningham, M. (1972). *Intelligence: Its Origins and Development.* New York: Academic Press.

Drescher, G. L. (1991). *Made-Up Minds*. Cambridge: MIT Press.

Droulez, J. & Berthoz, A. (1991). A neural network model of sensoritopic maps with predictive short-term memory properties. *Natl. Acad. Sci. USA*, 88: 9653-9657.

Duhamel, J. R., Colby, C. L., & Goldberg, M. E. (1992). The Updating of the Representation of Visual Space in Parietal Cortex by Intended Eye Movements. *Science*, 255: 90-92.

Ewert, J.-P. (1987). Neuroethology of releasing mechanisms: Prey-catching in toads. *Behav. Brain Sci.* 10, 337-405.

Grobstein, P. (1988). Between the retinotectal projection and directed movement: Topography of sensorimotor interface. *Brain Behav. Evol.* 31, 34-48.

Grobstein, P. (1992). Directed movement in the frog; motor choice, spatial representation, free will?. In *Neurobiology of Motor Programme Selection* (J. Kien, C. R. McCrohan, & W. Winlow, eds.). Pergamon Press: Oxford.

Hertz, J., Krogh, A. & Palmer, R. (1991). *Introduction to the theory of neural computation*. Addison Wesley: Redwood City (California).

Hummel, R. A., & Zucker, S. W. (1983) On the Foundations of Relaxation Labelling Processes, *IEEE Trans. Pattern Analysis and Machine Intelligence*, 5:267-287.

Ingle, D. (1976). Behavioral correlates of central visual function in Anurans. (*Frog Neurobiology*, R. Llinás and W. Precht, Eds), 435 - 451.

Ingle, D. (1983). Brain mechanisms of visual localization by frogs and toads. (*Advances in Vertebrate Neuroethology*, J. -P. Ewert, R. R. Capranica and D. J. Ingle, Eds), 177-226.

Liaw, J.-S. & Arbib, M. A. (1993) Neural Mechanisms Underlying Direction-Selective Avoidance Behavior. *Adaptive Behavior* , 1(3), 227-261.

Maes, P. & Brooks, R. A. (1990). Learning to coordinate behaviors. *AAAI-90*, Boston, 796-802.

Munkres, J. R. (1975). *Topology: a first course*. Prentice Hall.

Munoz, D. P., Pellison, D. & Guitton, D. (1991). Movement of neural activity on the superior colliculus motor map during gaze shifts. *Science*, 251: 1358-1360.

Piaget, J. (1954). *The Construction of Reality in the Child.* New York: Ballantine.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning internal representations by error propagation. In *Parallel distributed processing: Explorations in the microstructure of cognition* (D. E. Rumelhart, and J. McClelland, eds.), vol. 1, pp. 318-362. The MIT Press/Bradford Books.

Shen, W-M. (1994). *Autonomous Learning from the Environment.* New York: W. H. Freeman and Company.

Sokolov, E. N. (1975). The Neural Mechanisms of the Orienting Reflex. In *Neural Mechanisms of the orienting reflex*. (E. N. Sokolov and O. N. Vinogradova, eds.) Hillsdale: Lawrence Erlbaum Associates.

von der Malsburg, C. (1994). The Correlation Theory of Brain Function, (Reprint) In *Models of Neural Networks II* (E. Domany, J. L. van Hemmen, and K. Schulten, eds.), Ch. 2, pp. 95-119. Springer Verlag.

von der Malsburg, C. (1986). Am I Thinking Assemblies? In *Brain Theory*, G. Palm & A. Aertsen (Eds.). Springer-Berlag, Berlin.

von der Malsburg, C. (1988). Pattern recognition by labeled graph matching. *Neural Networks*, 1:141-148.

Emergent Hierarchical Control Structures: Learning Reactive/Hierarchical Relationships in Reinforcement Environments

Bruce L. Digney *
Defence Research Establishment Suffield
Box 4000, Medicine Hat, Alberta, CANADA, T1A 8K6
E_mail bdigney@dres.dnd.ca

Abstract

The use of externally imposed hierarchical structures to reduce the complexity of learning control is common. However, it is acknowledged that learning the hierarchical structure itself is an important step towards more general (learning of many things as required) and less bounded (learning of a single thing as specified) learning. Presented in this paper is a reinforcement learning algorithm called Nested Q-learning that generates a hierarchical control structure in reinforcement learning domains. The emergent structure combined with learned bottom-up reactive reactions results in a reactive hierarchical control system. Effectively, the learned hierarchy decomposes what would otherwise be a monolithic evaluation function into many smaller evaluation functions that can be recombined without the loss of previously learned information.

1 Introduction

The use of Q-learning [1] and other related reinforcement learning techniques is common for the control of autonomous robots [2]. However, long learning times combined with the slow speed (when compared to simulations) and the frailties of robot hardware present considerable problems when scaling up to real applications. Often, to reduce the problem to a more tractable size, the control problem is hand decomposed into a hierarchical structure [3]. This abstracts it into many smaller, more easily learned portions. However, hand decomposition imposes the designer's preconceived notions on the robot which, from the robot's point of view, may be inefficient or incorrect. Furthermore, it is acknowledged that for truly general learning and full autonomy to occur in the face of unknown and changing environments, the structure of the hierarchical control system must also be learned.

Presented in this paper is the Nested Q-learning algorithm that allows the generation of hierarchical control structures in reinforcement learning domains. Once the structure has been learned, skills that have been previously mastered can be used in future tasks and environments. This continual carrying forward of learned information is called life-long learning and provides the robot with a head start when learning new tasks [4]. As the robot moves from task to task and environment to environment it will have the accumulated information of its past experiences available to it as skills. These transportable skills will allow the robot to learn progressively more complex tasks. Eventually this continual learning will allow the robot to learn tasks that would be impossible in a simple monolithic network. Having the robot learn a hierarchical control structure is analogous to a student who invests initial effort in discovering the underlying principles or structure of a problem, rather than simply memorizing a monolithic solution. Once the underlying principles are understood they can be transferred to more difficult tasks. Such is not possible if solutions are only memorized, making the information gained albeit at a lesser expense, useless in new situations. This continual learning lends itself to use in pre-training or shaping, either by chance or through a regimented training program. In nested Q-Learning controlled robots the use of scaffolding actions and staged learning to exploit this online skill transfer between task/environment settings, is discussed in more detail elsewhere [4].

2 Learning Hierarchical Structures

2.1 Introduction

Consider the example of a robot perceiving its world through sensors and receiving internal and external reinforcement as pictured schematically in Figure 1(a). This robot is capable of acting on its world using a number of primitive actuator movements. These primitive actions are at the simplest level of the robot's actuators and although they may utilize feedback control mechanisms, they do not embody any higher intelligence. The robot also receives reinforcement signals which are critical indicators of how the robot is progressing with respect to the completion of some desired task(s). These critical signals are all the direction that can be assumed for the autonomous robot. Critical error signals simply provide negative reinforcement when the actions of the robot do

* The author acknowledges support from the Canadian Department of National Defence (DND) and the Natural Sciences and Engineering Research Council (NSERC).

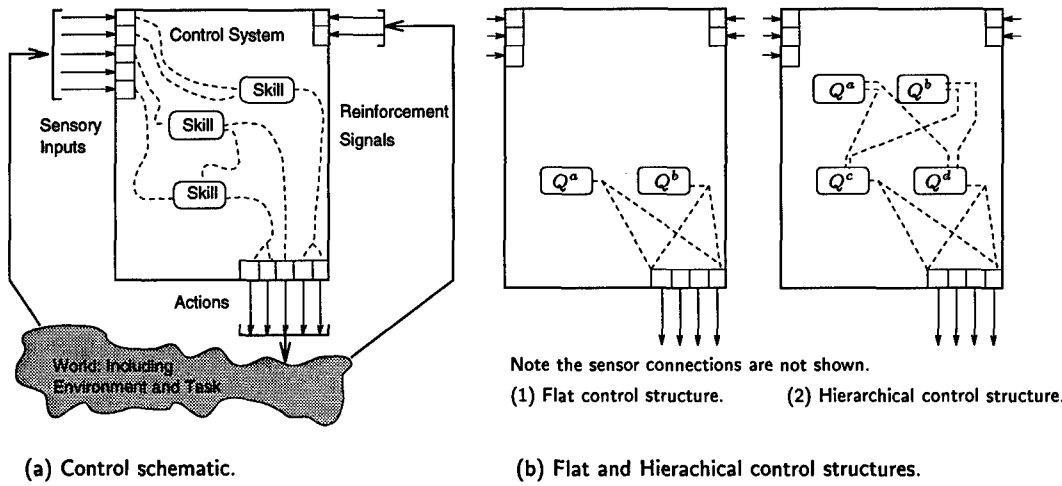


Figure 1: Schematic of control architectures: (a) sensory, action and reinforcement signal configuration for a learning control system and (b) control architectures: (1) flat structure and (2) hierarchical structure.

not achieve the goal and positive (favorable) reinforcement whenever the actions achieve the goal.

In the field of intelligent control, the strategies connecting the sensors to the actions are either hardwired by a designer, taught to the agent or left to be autonomously learned by the agent. There are many variations [5] [6] of architectures in which such control systems may be implemented. The two main variations are flat and hierarchical as shown schematically in Figure 1(b), (1) and (2), respectively. Note that the hierarchical control system relies upon higher level skills being built upon lower level skills while the flat architecture contains only skills at a single level which interact directly with the actuators.

2.2 *Related Work*

Many researchers have recognized the need for hierarchical structures in learning control systems. Research most closely related to the approach described later in the paper will be briefly summarized.

Hierarchical learning control systems are often handcrafted with a number of low level behaviors controlled by a gating function. The low level behaviors map the current state of the agent into actuator activities while the gating function determines which low level behavior is to be active. Maes and Brooks [7] used this approach by hand-crafting the individual low level behaviors while the gating system was learned from a reinforcement signal during operation. Mahadevan and Connell [8] used the complementary approach in which a hand-crafted gating function and separate reinforcement signals which were supplied for each individual behavior. Lin [9] used a similar approach combined with staged learning to first train the individual behaviors and then train a gating function. Dayan and Hinton's Feudal Q-learning [10] used a hierarchical master/slave type architecture with a set of hand-crafted commands and corresponding reinforcement functions. The commands served as actions available to the high level master. As the master selected a command, a low level slave was tasked with its performance and was subsequently rewarded for the performance of the command regardless of whether or not it achieved the overall task. On the other hand, the master was rewarded only for the completion of the overall task. In this approach, learning occurs at both levels simultaneously. The lower levels learn the commands and the higher level learns when to invoke the low level commands to perform some desired task. Singh's Compositional Q-learning system[11] first learned elemental tasks and then the gating functions that switch between elemental tasks in the correct temporal sequence to achieve some high level goal. Kaelbling's Hierarchical Distance to Goal (HDG) [12] viewed the world on two levels of resolution. Regions, with their centres referred to as landmarks, were used to move an agent into the same region as the goal's location. Then local actions were used to move to the goal.

These approaches are all similar in the respect that the structures are hand designed and individual components are learned. In the nested Q-learning algorithm described in this paper, it is proposed that the structure itself be learned. The structure that emerges can be of as many levels as the task requires and not just two levels. When an agent initially starts out, no information about structure or useful behaviors is given to it. The agent discovers distant recognizable features in its world and learns the relationship between different features, its environment and its tasks. These relationships form skills which are usually a hierarchical assembly of other skills and primitive actions. Currently, features are restricted to being evenly distributed over raw sensor signals, but work to allow the discovery of higher level compound features is being pursued. Using nested Q-learning, the agent also learns bottom-up reactive/opportunistic functions which serve to 1) provide a high level command source at the top of the hierarchy and 2) facilitate the invocation of previously learned beneficial behaviors in new situations without the necessity of relearning them. Current work is extending the concept of bottom-up functions to include not only the learning of reactive/opportunistic situations, but the

conditions under which skills are applicable, as well.

In summary, the major advances of nested Q-learning over the described approaches are: 1) the ability to autonomously construct hierarchical structures of arbitrary depth from sensory and reinforcement signals, 2) learning occurs at all levels (converging form the actuators upward) and 3) the use of learned bottom-up reactive functions allows skills to invoke themselves and not necessarily require an invoking signal form another skill. These reactive functions serve to control the highest level of the hierarchy and allow for previously learned skills, of immediate benefit or danger, to be transfered between tasks and environments.

2.3 *Nested Q-learning*

A method will now be presented that will allow for the autonomous generation of hierarchical control structures as pictured in Figure 1(b)(1). Each incoming sensor has a finite number of perceivable distinct values. These distinct values are referred to as *features*. For example, consider a robot whose task it is to move between two rooms through an open door. In this task, the first feature or sensory condition the robot needs to perceive is the door directly ahead. The next relevant feature would be for the robot to perceive the door frame on either side of it. The last relevant feature would be for the robot to perceive the door directly behind it. Eventually control strategies will be learned to move the robot so as it can perceive these features. Current work will allow such complex compound features, such as doorways, to be recognized, but in this paper a simple even distribution of features over the range of each sensor will be assumed. For sensor s_n these features are represented by the distinct values

$$s_n \in \{ \; s_n^1, \;\; s_n^2, \cdots \;\; s_n^m, \;\; \cdots \;\; s_n^{M_n} \; \} \qquad (1)$$

where s_n is the n^{th} sensor, s_n^m, is the m^{th} distinct value and M_n is the number of distinct values for sensor n. For any number of incoming sensors their distinct values or recognizable sensory conditions will constitute *features* which may or may not prove useful in controlling the agent. These features, label f_1 through f_I, are defined over all distinctive values (1 to M_n) for all sensors (1 to N),

$$f_i \in \{ \; s_1^1 \cdots \;\; s_1^{M_1}, \cdots \;\; s_n^{m_n}, \cdots \;\; \cdots \;\; s_N^{M_N} \; \} \qquad (2)$$

where f_i is the i^{th} distinct feature and N is the number of sensors. The total number of features is, $I = \sum_{n=1}^{N} M^n$.

A Q-learning evaluation function is now defined for each distinct feature, Q^{f_i}. In this method, with features defined as distinct recognizable sensory conditions, skills become the control strategies required to cause the robot to attain those particular sensory conditions. Eventually the lowest level primitive actions must be invoked by the learned control strategies. These primitive actions are the simple low level actuator movements with which the robot acts on its world. They may contain some form of feedback control mechanism, but remain the simple building blocks out of which complex control strategies (skills) can be constructed. There are J primitive actions, labelled a_1 through a_J.

Each feature, f_i, is by definition the endpoint of a skill with its control strategy represented by the evolving evaluation function, Q^{f_i}. While attempting to reach some desired feature, $f_{desired}$, the control strategy can invoke any of the primitive actions, a_j (shown as Q^{a_j}) or any of the skills represented by all Q^{f_i}. The choice of possible actions, u, is now

$$u \in \{ \; Q^{a_1}, \;\; \cdots Q^{a_J}, \;\; Q^{f_1}, \;\; \cdots \;\; Q^{f_I} \; \} \qquad (3)$$

where u is a possible action or skill choice, Q^{a_j} is a non-adaptive primitive action (shown with similar notation to the Q functions for convenience) and Q^{f_i} is an adaptive skill. The total number of possible actions is the sum of all primitive actions and all currently possible skills, $u_{total} = J + I$. The state x_l of the agent is established by the state of all the incoming sensors, x_1 through x_L,

$$x_l \in \{ \; x_1, \;\; x_2, \;\; \cdots \;\; x_l \;\; \cdots \;\; x_L \; \} \qquad (4)$$

where x_l is a distinct state of the agent. The evaluation function for each feature is a function of the robots current state and all possible actions, $Q^{f_i} = f(x, u)$. The evaluation function for each skill now becomes,

$$Q^{f_i} = f(x_1, \cdots x_L, Q^{a_1} \cdots Q^{a_J}, Q^{f_1}, \cdots Q^{f_I}) \qquad (5)$$

where both the number of states, L, and the number of skills, I, are open ended and subject to initial discovery and then to increases or decreases due to ongoing changes. It is seen that the learning algorithm described above becomes nested and possibly recursive. That is, the evaluation function, $Q^{f_{desired}}$, can invoke other skills including itself while attempting to reach feature $f_{desired}$. It is this nested nature that will allow hierarchical control structures to emerge.

As the agent interacts with the environment it receives an external reinforcement signal(s), r_{EXT}. It is through these signals that the agent is driven to perform tasks of external benefit. This reinforcement signal is defined as

$$r_{EXT} = \begin{cases} 0 & \text{if external task is achieved} \\ -R_{\text{EXT}} & \text{otherwise} \end{cases} \qquad (6)$$

where r_{EXT} is an external reinforcement signal and R_{EXT} is a positive constant. In addition, there are various internal reinforcement signals, r_{INT}. These drive the agent to perform tasks of internal benefit such as *avoid danger* and *find fuel*. In the nested Q-learning algorithm there is also a reinforcement signal that effects only the currently active skill(s). This reinforcement signal drives the action of the agent to reach the desired feature.

$$r_{\text{FEAT}} = \begin{cases} 0 & \text{if at desired feature} \\ -R_{\text{FEAT}} & \text{otherwise} \end{cases} \qquad (7)$$

where r_{FEAT} is the feature's reinforcement signal and R_{FEAT} is a positive constant.

For the top-down goal directed action/skill selection the action chosen, u^*, is determined by

$$u^* \Leftarrow \begin{cases} \text{argmax}_u\{Q^{f_i \text{ or } a_j} + E\} & \text{if } Q^{f_i} \text{ active} \\ a_j & \text{if } Q^{a_j} \text{ active} \end{cases} \quad (8)$$

where argmax_u is the maximum function taken over all possible primitive actions and skills, Q^{f_i} is an adaptive skill (an evolving function of the state x and all possible actions u), Q^{a_j} is a primitive action, a_j is some specific action to be taken and E is the exploration strategy. The nested nature is seen again in Equation 8. The currently active skill can select another skill or a primitive action. If another skill is selected that skill can go on and select yet anther skill and so on. If a primitive action is selected, Q^{a_j} then the physical actuator action a_j is performed. The exploration component, E, of Equation 8 is required to ensure adequate exploratory coverage of the agent's state space. It can be random, error or recency based. In this development a recency based exploration policy is used which ensures that the actions (primitive/skill) that were taken less recently are favored over the more recent actions [13].

Upon performing the selected action, u^*, be it a primitive action or a skill, the robot advances from state x_v to the next state x_w and incurs a total reinforcement signal, r_{TOTAL}. Included in this total reinforcement signal is the cost of performing the selected action. This cost includes the physical cost of performing the action and possibly the mental (computational) cost of choosing the action. These costs are designated as r_{LOW}, with

$$r_{\text{LOW}} = \begin{cases} -C & \text{if } u^* \text{ is a primitive action} \\ \sum_{k=0}^{K} r^{u*}_{\text{TOTAL}}(k) & \text{if } u^* \text{ is an adaptive skill} \end{cases} \quad (9)$$

where $\sum_{k=0}^{K} r^{u*}_{\text{TOTAL}}(k)$ is the total reinforcement signal from the invoked skill summed over the number of steps, K, required to perform the skill, u^*, and C is a constant that reflects the cost of performing the primitive action. The total reinforcement signal for the invoking skill becomes

$$r_{\text{TOTAL}} = r_{\text{EXT}} + r_{\text{INT}} + r_{\text{FEAT}} + r_{\text{LOW}} \quad (10)$$

This total reinforcement is used to construct the Q functions of expected reinforcement, from which useful top-down control strategies will emerge. The error, e_Q, is defined to be,

$$e_Q = \gamma \cdot \max_u\{Q_{x_w,u}\} - Q_{x_v,u^*} + r_{\text{TOTAL}} \quad (11)$$

where γ is the temporal discount factor $0 < \gamma < 1$ and $\max_u\{Q_{x_w,u}\}$ is the current prediction of the maximum total future reinforcement remaining when the agent leaves state x_w. This error is used to adapt the evaluation functions,

$$Q_{x_v,u=u^*}(k+1) = Q_{x_v,u=u^*}(k) + \eta_Q \cdot e_Q \quad (12)$$

$$Q_{x_v,u\neq u^*}(k+1) = Q_{x_v,u\neq u^*}(k) \quad (13)$$

where η_Q is the rate of adaptation and k is the index of adaptation.

The preceding derivation describes a nested Q-learning technique through which a top-down action selection mechanism will generate a hierarchical control structure and propagate goal seeking commands downward though it. Each skill, whenever invoked, will in turn invoke other skills and/or primitive actions in an attempt to fulfil the desired goals of higher skills. The bottom-up or sensory based action selection mechanism and how it interacts with the top down action selection mechanism will now be described. This bottom-up selection mechanism will allow skills to invoke themselves without the need of a top-down command signal from a higher level skill. Consider the agent at state $x_{invoked}$ when the skill Q^{f_i} is invoked. Upon the arrival at the sensory conditions of the defining feature f_i, two things can occur: 1) an uneventful arrival at f_i with only nominal reinforcements occurring, or 2) arrival at f_i coincides with a markedly high negative or high positive reinforcement signal. Such a distinct change in reinforcement would represent possible reactive or opportunistic behaviors. The invocation of the selected skill (and in turn all other sub-skills and actions) from any other state may not necessarily result in such non-typical results or useful correlation. For instance, the invocation of the *avoid obstacle* skill would not yield any benefit if an obstacle was not present. Similarly, the invocation of the *begin feeding* skill would not be of any benefit if food was not near. The sensory conditions that are a precursor to reactive/opportunistic situations must be learned. In the first example, the sensory situations that indicate the presence of an obstacle must be learned in order to activate the avoid obstacle behavior when it is useful. By learning these situations in a bottom-up manner (outside of the top-down Q^{f_i} functions) it will become possible to have skills invoke themselves in new situations where the top-down functions have not yet been formed. By allowing skills to invoke themselves or make themselves more likely to be invoked will speed the learning of new, but related, tasks.

For example, consider a robot learning to deliver the mail. During the performance of this task it learns that it requires intermittent recharging. It soon learns how to recharge itself and also learns which sensory conditions are associated with an opportunity to recharge; possibly the appearance of a recharge outlet. In this task, recharging will be learned from scratch as a particular skill, composed of a structure of many sub-skills and actions. As the agent learns the recharging skill, a bottom-up activation function is formed for that skill as well. In the presence a fortuitous recharging outlet where there was not one before, the bottom-up function will respond strongly and attempt to override any top-down commands that might be active. Moreover, when the robot is set to learning a different task (e.g. sweep the floor) these bottom-up activations will remain valid. Now, if the robot comes across a recharging outlet at its new task, it will know how to respond, and know the proper actions to respond with, without having to relearn the recharging skill.

To implement these bottom-up reactive relationships an evolving function, $B^{f_i}(x_{invoked})$, is defined for each skill Q^{f_i}. This function learns to predict the reinforcement outcomes of invoking each skill from differ-

ent states. Effectively, these functions evolve into the bottom-up triggering response for reactive and opportunistic skills. Within an emergent hierarchy, such a bottom-up action selection mechanism will be in control at the very top of the hierarchy as there are no higher levels to invoke them in a top-down manner. They will also be able to subsume the current top-down flow of commands should a reactive or opportunistic situation present itself outside of the converged top-down regimented control strategies. If a bottom-up triggering situation presents itself with enough regularity, it will eventually be absorbed by the top-down control strategies. Bottom-up reactive behaviors are most valuable when the agent is in new situations where the top-down structure has not yet formed. They represent chunks of control structure and the skills within that were learned in the past that might be useful in new task/environment settings. Useful skills can be invoked in new situations without the need for the robot to relearn them. As these skills usually contain their own top-down structures, the bottom-up contributions will be important in transferring information between tasks and environments.

The error, e_B, for these functions is determined using the eventual reinforcement, r_{EVT}, that occurs at the defining feature.

$$e_B = r_{EVT} - B^{f_i}(x_{invoked}) \tag{14}$$

where $B^{f_i}(x_{invoked})$ is the bottom-up reactive function for skill Q^{f_i}, $x_{invoked}$ is the state from which Q^{f_i} was invoked and r_{EVT} is the total reinforcement when the feature has been reached. The reactive function is then adapted,

$$B^{f_i}(x_{invoked})(k+1) = B^{f_i}(x_{invoked})(k) + \eta_B \cdot e_B \tag{15}$$

where η_B is the learning rate. Eventually the reactive functions, B^{f_i}, will learn what sensory conditions the invocation of skill Q^{f_i} will be beneficial. Furthermore, the B^{f_i} functions learn which sensory signals that are relevant and which are irrelevant and can be ignored. When B^{f_i} is included in the top-down action selection mechanism of Equation 8, it will favor the selection of proven opportunistic and reactive skills above others, even those actions that it has learned to select. In the current example, if the recharging boxes are marked by a red light, it will be learned that the presence of a red light signal will trigger the recharge skill while all other extraneous sensory signals (such as spatial location and floor color) will be ignored.

3 Simulation

To evaluate nested Q-learning the simple two dimensional animat and world of Figure 2 was used. The animat's primitive actions were capable of moving it to one of four adjacent spatial locations. The animat's sensors perceived the color of the floor panel below it, the color of an overhead signalling light and the animat's spatial location within the world. The world is shown in Figure 2(c). It had white colored floor panels except for one blue and one green floor panel at the locations indicated. It was possible for the location of these blue and green panels to change, moving to locations indicated by either $World_1$ or $World_2$ in Figure 2(c). The animat remained unaware of these changes as it could only sense what was happening at its current spatial location within the world. Tasks for the animat were externally specified using reinforcement signals and could entail anything from movement to a desired spatial location to matching sensed floor panel color with the color of the signalling light. Summarized in Figure 3 are the features found to be relevant by the animat. Although these features were subject to random discovery, they are presented in an orderly list for the readers benefit in the following analysis. Shown in Figure 4 are the sensory, motor and reinforcement connections for the animat. The incoming sensors are the light color, S_{light}, the floor color, S_{floor}, and the spatial location, $S_{spatial}$. The outgoing actuator commands are up, Q^0, down, Q^1, left, Q^2, right, Q^3 and stay, Q^4. The incoming reinforcement signal is r_{EXT}. The initial control system is shown as a random structure of unknown skills, $Q^?$. During these tests the learning rates, η_Q, η_B, and the temporal discount factor, γ, were set to 0.2, 0.01 and 0.9, respectively.

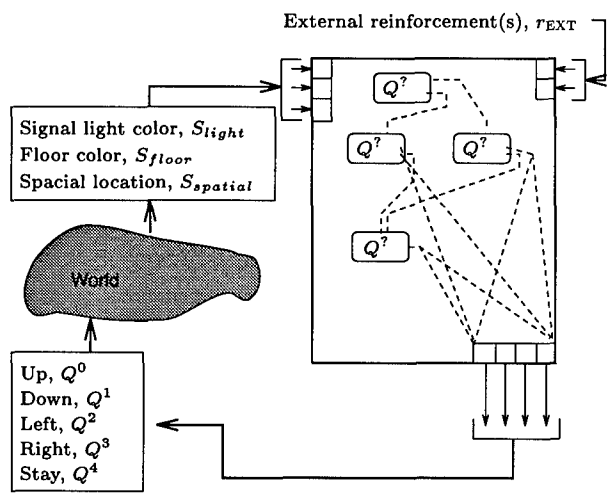


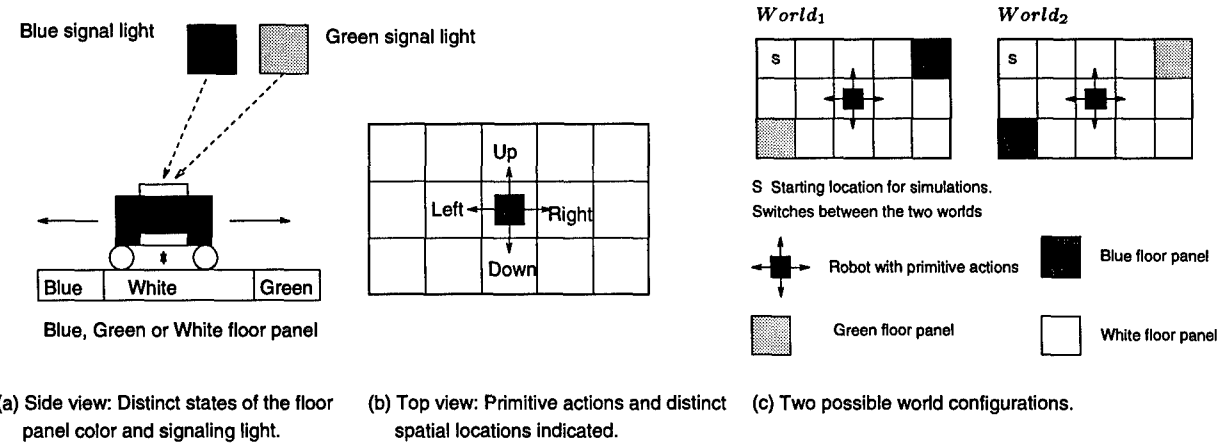
Figure 4: Connections of sensors, actuator and reinforcement signals and initial random structure of control system. For clarity the sensor connection are not shown.

3.1 Generation of Control Hierarchies

To evaluate the capabilities of nested Q-learning to generate control hierarchies the agent was rewarded with the external reinforcement signal of Equation 16.

$$r_{\text{EXT}} = \begin{cases} +2 & \text{if light is blue and floor is blue.} \\ +4 & \text{if light is green and floor is green.} \\ -1 & \text{otherwise.} \end{cases} \tag{16}$$

The locations of the blue and green floor panels were set randomly switching between the two possible configurations as shown in in Figure 2 (c). The signal light was set alternating between blue and green. An animat with no prespecified information was placed in this



(a) Side view: Distinct states of the floor panel color and signaling light.

(b) Top view: Primitive actions and distinct spatial locations indicated.

(c) Two possible world configurations.

Figure 2: Simulated animat's sensory systems: (a) Floor color and signal light sensors, (b) distinct spatial locations and possible movements of the animat. Simulated world: (c) Blue and green floor panels in two possible configurations as indicated. Locations of the blue and green floor panels can change to that of $World_1$ or that of $World_2$.

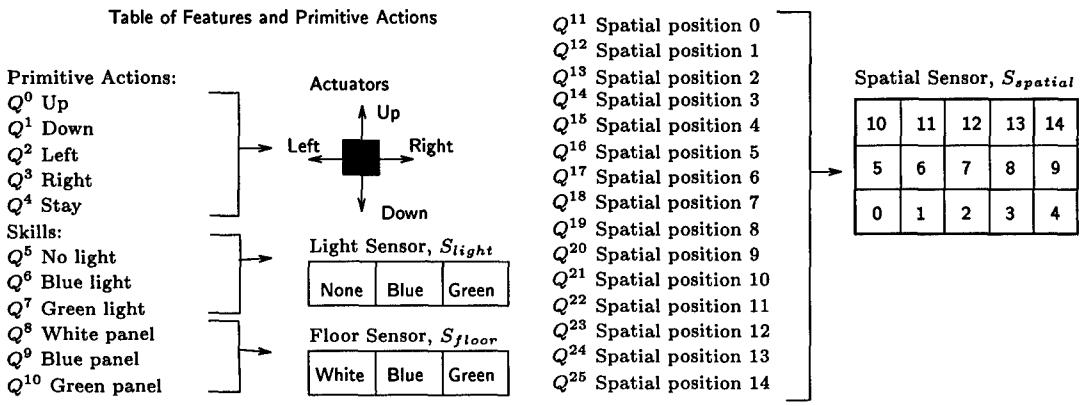


Figure 3: Summary of skills and features.

world and allowed to learn how to maximize its rewards over time. Performance plots for all skills and primitive actions versus time are presented in Figure 5. The vertical axis shows the performance of each skill or primitive action. Performance is taken to be the total reinforcement signal that the skill responds with whenever it is invoked. The axis pointing out of the page is the number of the skill or action and the horizontal axis represents the expired time. For clarity, selected skills have been extracted from the plot in Figure 5 and shown individually in Figure 6. From these figures it is seen that the primitive actions responded consistently with a performance of -1.0 (as they should), while all the adaptive skills performed changed over time and usually improved (if indeed they could). The first skills mastered were the ones defined by spatial locations, for example skills Q^{11} and Q^{25}. The two skills which proved to be higher level skills, Q^9, *find blue panel* and Q^{10}, *find green panel* were subsequently mastered using the two skills of Q^{11} and Q^{25}.

Sensations or features over which the animat had no control were never learned (nor could they be). For instance, Q^5, *find no signal light* could never be learned, as the signal light was always either blue or green and was never off. Interestingly enough, it was originally expected that a similar situation would occur for the other two signal light related behaviors, *find green signal* while the blue light was on and *find blue signal* while the green light was on, because the lights were not directly controlled by the agent. However, the light was alternating blue, then green, then blue, etc., changing upon task completion. The animat soon discovered that if it wanted to see a green signal light, all it had to do was go to the blue panel and once the task triggered by the blue light was completed, the green light would come on in place of the blue light. Instances of the agent finding novel solutions and taking advantage of unintentional loopholes in the simulation were common during these trials.

The bottom-up responses for all the possible skills are shown in Figure 7. Figure 7(a) shows the bottom-up response for an invoking state in which the blue signal light is on. Other details of that state were discovered by the control system to be irrelevant, as would be expected. Figure 7(b) shows the bottom-up response for an invoking state in which the green signal light was on. As indicated, the predominant bottom-up response for the green signal corresponded to Q^{10}, or *find the green panel*

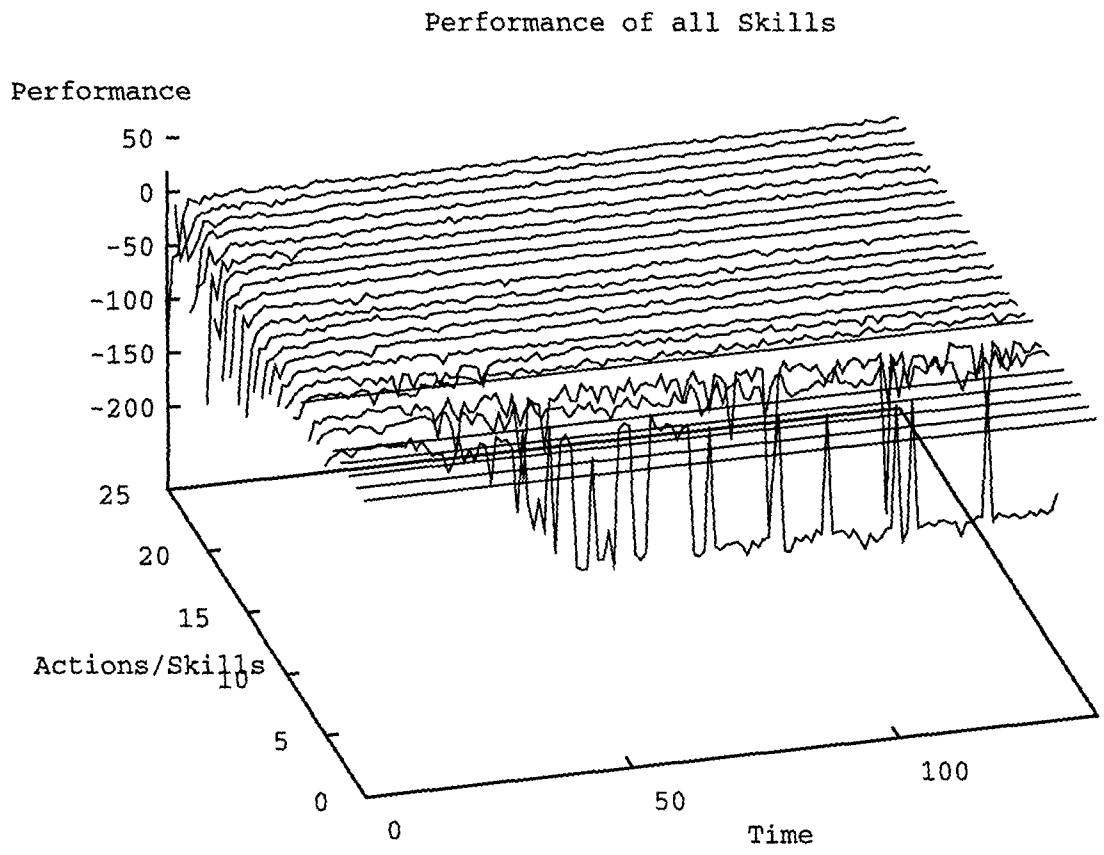
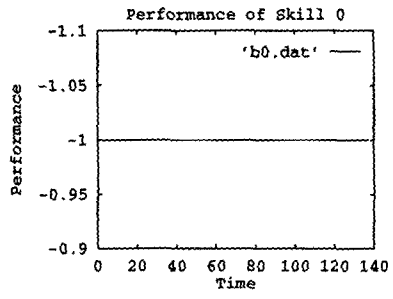
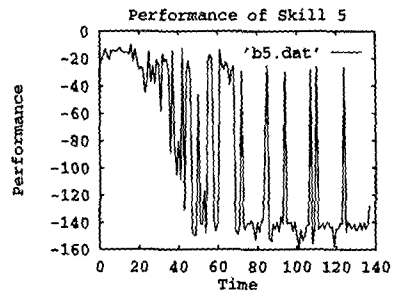


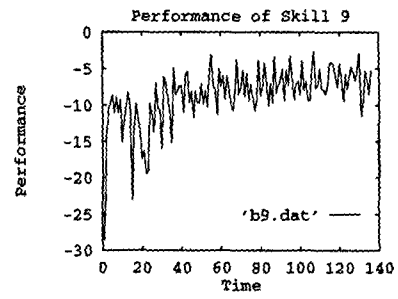
Figure 5: The performance of all skills.



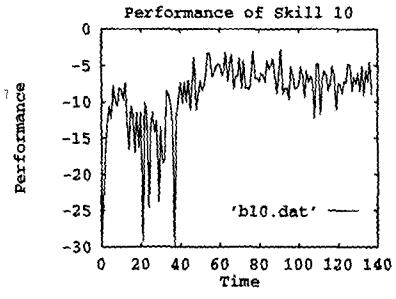
(a) Peformance for action Q^0, (up).



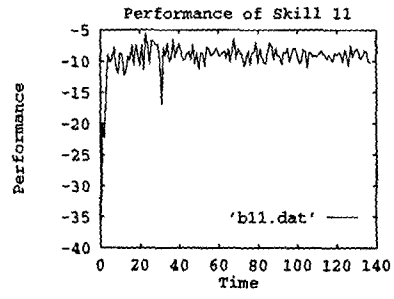
(b) Peformance of skill Q^5, (no light).



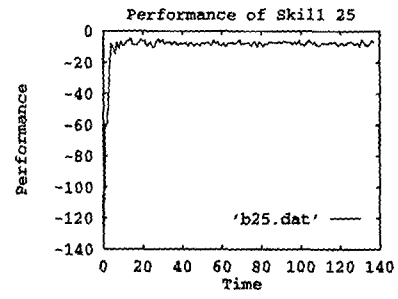
(c) Peformance of skill Q^9, (blue panel).



(d) Peformance of skill Q^{10}, (green panel).



(e) Peformance of skill Q^{11}, (spatial 0).



(f) Peformance of skill Q^{25}, (spatial 14).

Figure 6: Detailed performance plots for selected skills.

and for the blue signal it corresponded to Q^9, or *find the blue panel.* These bottom-up responses were shown to respond high for the skill that was capable of fulfilling the task that would result in favorable reinforcement signals. In these simulations, the green light triggered skill Q^{10} and the blue light triggered Q^9. These high level skill then invoked other skills and eventually primitive actions in a top-down manner to fulfil the tasks.

The control strategies invoked by the bottom-up reactions of Q^9 and Q^{10} are shown in Table 1 and Table 2. The schematic shown in Figure 8(a) shows the distinct two level architecture that emerged from the simulation. In Tables 1 and 2, the starting location is taken to be the upper left corner of the world at spatial position 10 as indicated in Figure 2(c). Tables 1 and 2 correspond to the configuration of the world as $World_1$. When the world is configured as $World_2$, Q^9 and Q^{10} are represented by Tables 3 and 4, respectively. Figure 8(b) shows the movements of the animat for the skill, Q^9 for both world configurations. It is clear that a hierarchical control system emerged which, when invoked by a bottom-up opportunistic drive, began to search for the locations of the correctly colored panels. It should be noted that as this search required two way movement through the state space and the continual disruption of a single monolithic evaluation function, learning this problem in a non-hierarchical architecture would be difficult. That is, for the current task, some states in a monolithic evaluation function would have a constantly changing correct action. Such a moving target would make convergence impossible. By decomposing the task into a hierarchical assembly of skills, the evaluation functions that represented the skills had a constant target to converge to.

Table 1: Skill Q^{10}: Find green floor panel (for $World_1$ with starting point as indicated in Figure 2(c)). Note: the length of the arrows indicates the depth of the command signal into the hierarchy. For example, $->$ indicates that the command signal is at the top of the hierarchy, while $--->$ indicates that the control signal is down two levels into the hierarchy.

Level Down	Command	Comments
1	$-> Q^{10}$	Find green panel
2	$--> Q^{11}$	Spatial location 0
3	$---> Q^{1}$	Down
3	$---> Q^{1}$	Down and at goal

3.2 *Generation of Novel Hierarchies*

Often as the structure emerged without direct input from the designer, many unexpected but successful structures emerged. For example, in a related simulation, the agent was required to learn routes to spatial positions 4 and 14, represented by skills Q^{15} and Q^{25}, respectively. For some reason, skill Q^{20}, the skill to spatial position 9, which lies directly between positions 4 and 14, converged faster that the other skills. Skills Q^{15} and Q^{25} then learned to use Q^{20} to arrive close to their goal and then

Table 2: Skill Q^9: Find blue floor panel (for $World_1$ and starting point as indicated in Figure 2(c)).

Level Down	Command	Comments
1	$-> Q^{9}$	Find blue panel
2	$--> Q^{11}$	Spatial location 0
3	$---> Q^{1}$	Down
3	$---> Q^{1}$	Down
2	$--> Q^{25}$	Spatial location 14
3	$---> Q^{3}$	Right
3	$---> Q^{3}$	Right
3	$---> Q^{3}$	Right
3	$---> Q^{3}$	Right
3	$---> Q^{0}$	Up
3	$---> Q^{0}$	Up and at goal

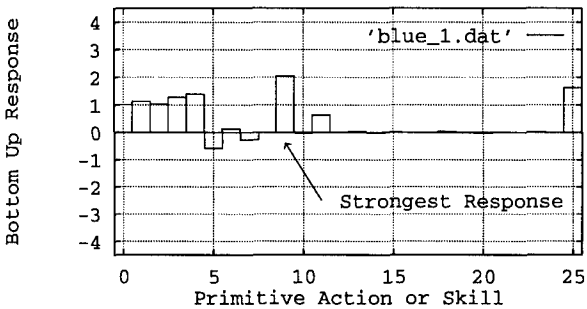
Table 3: Skill Q^{10}: Find green floor panel (for $World_2$ and starting point as indicated in Figure 2(c)).

Level Down	Command	Comments
1	$-> Q^{10}$	Find green panel
2	$--> Q^{11}$	Spatial location 0
3	$---> Q^{1}$	Down
3	$---> Q^{1}$	Down
2	$--> Q^{25}$	Spatial location 14
3	$---> Q^{3}$	Right
3	$---> Q^{3}$	Right
3	$---> Q^{3}$	Right
3	$---> Q^{3}$	Right
3	$---> Q^{0}$	Up
3	$---> Q^{0}$	Up and at goal

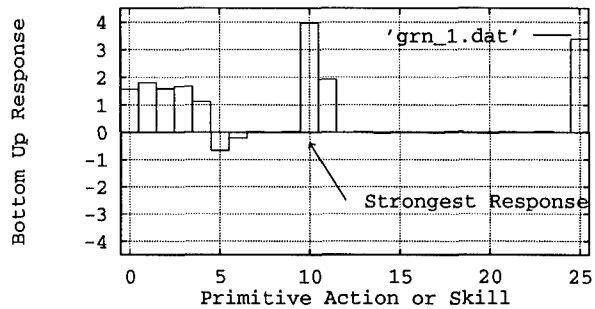
it was a single step to reach the desired location. This hierarchical control structure is illustrated in Figure 9(a) with the animats actions in Figure 9(b). Throughout the course of these simulations many other examples of interesting multi-level hierarchies emerged, each with the agent developing some unique and innovative method of solving the control problem.

3.3 *Information Transfer Between Tasks*

In this simulation the world was again set to alternating between its two possible states. Initially, the desired task, as defined by Equation 17, was to find the blue floor panel whenever the blue light was on. A hierarchy using the two skills Q^{11} and Q^{25} emerged to ultimately perform Q^9 (find blue panel) as shown in Figure 10(a). Next, a new task was requested as per the external reinforcement signal of Equation 18. The new task was to find the green panel whenever the green light was on. With most of the control strategy already learned from the previous task, the animat now had to learn how to utilize these existing skills to perform the new task. This was accomplished by learning skill Q^{10} (find green panel), extending the hierarchy to that pictured in Figure 10(b).



(a) Bottom up reactive response for the blue signal light on. Note the strength of response for skill Q^9.



(b) Bottom up reactive response for the blue signal light on. Note the strength of response for skill Q^{10}.

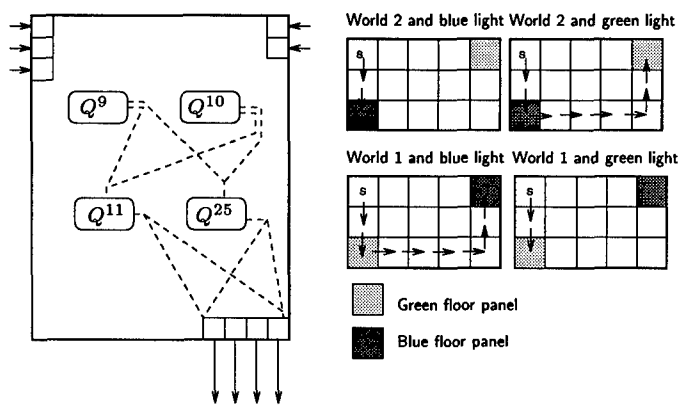
Figure 7: Bottom up reactive responses for all skills and primitive actions for sensory conditions of (a) blue signal light on and (b) green signal light on.

Table 4: Skill Q^9: Find blue floor panel (for $World_2$ and starting point as indicated in Figure 2(c)).

Level Down	Command	Comments
1	$-> Q^9$	Find blue panel
2	$--> Q^{11}$	Spatial location 0
3	$---> Q^1$	Down
3	$---> Q^1$	Down and at goal

$$r_{EXT} = \begin{cases} +4 & \text{if light is blue and floor is blue.} \\ -1 & \text{otherwise.} \end{cases} \quad (17)$$

$$r_{EXT} = \begin{cases} +4 & \text{if light is blue and floor is blue.} \\ +4 & \text{if light is green and floor is green.} \\ -1 & \text{otherwise.} \end{cases} \quad (18)$$



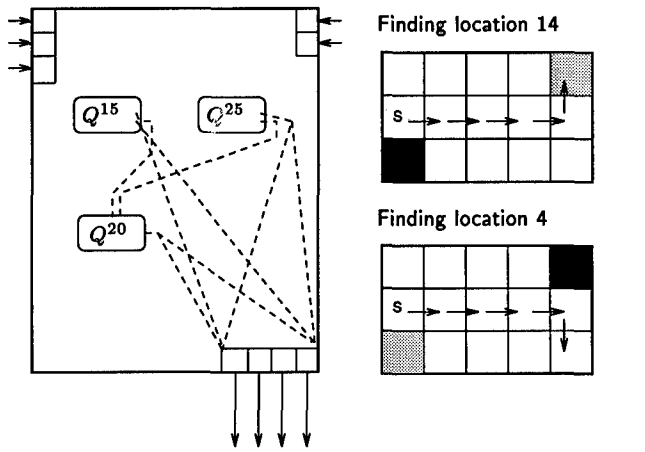
(a) Emergent structure. (b) Actions taken by the robot.

Figure 8: Generation of a hierarchical control system. (a) the structure that emerged with two bottom-up driven skills, Q^9 and Q^{10}, at the top, (b) actions taken by the animat.

4 Discussion

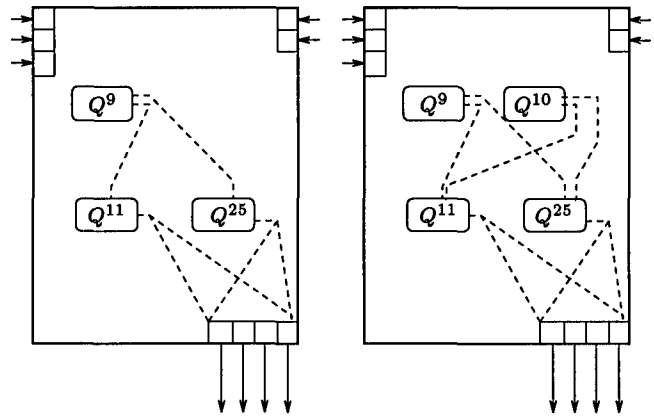
The hierarchical structures outlined emerged as sensory-motor relationships and converged from the bottom (closest to the actuators) upward. Those sensor-motor skills that required only primitive actuator movements were discovered and learned first. Higher level skills could then be built upon lower level skills. Once the low level spatial skills (Q^{11} and Q^{25}) converged, or at least had begun to be performed reliably, then the higher level skills (Q^9 and Q^{10}) could be learned. The learning of these stratified skills required both the learning of temporal sequences of primitive actions and/or skills as well as which sensor information was relevant or irrelevant at each emerging level of behavior. In these simulations the agent learned that the skills involved with finding spatial locations were independent of the color of the signal light and floor panels was irrelevant and also that the agent needed only primitive actions to perform reliably. This was not always necessarily true. Although only primitive actions were needed, this did not prevent the emergence of convenient hierarchical structures. An example of the learning system finding novel convenient methods of solving its problems was presented in Section 3.2 in which an intermediate location was used to find adjacent spatial locations.

Although initially necessary, the exploration component later contributes to poorer performance. Within such an evolving control system, it might be desired that skills crystallize, that is, limit and eventually cease the exploration component of the skill, once exploration begins to prove fruitless. This would result in skills becoming stratified from the actuators upward, freeing limited computation resources for discovering and learning yet higher skills. Should changes or malfunctions occur, the exploration component would be reactivated in the effected section of the control hierarchy. As the skills converged, many sensor inputs proved irrelevant to the evaluation and reactive functions and could be removed. For example, the skills necessary to find spatial locations were independent of signal light color and floor panel color and relied only upon a sense of location. Current



(a) Emergent structure. (b) Actions taken by the robot.

Figure 9: The hierarchical control system that emerged for skills, Q^{15} and Q^{25}, utilizing the intermediate skill Q^{20}. (a) the structure that emerged, (b) actions taken by the animat.



(a) First skills learned Q^9 Q^{11} and Q^{25}. (b) Second skill learned, Q^{10}.

Figure 10: Transfer of skills between tasks: (a) control structure after the first task, fulfilled by skill Q^9, is learned, and (b) after the second task, fulfilled by skill Q^{10}, is learned. Note: Q^{10} learns to use previously acquired skills.

work will allow the agent to prune sensory and action connections from skills when they are discovered irrelevant. Again, this would further crystallize the skills into encapsulated and efficient sensor-motor control systems. Another extension that becomes evident is that the abstraction of features must be allowed. Just as the control strategies are built from abstractions of primitive actions and lower skills, features must be allowed to construct hierarchies of abstraction.

5 Conclusions

The nested Q-learning technique developed in this paper generated hierarchical control systems for the control of a simple simulated animat. These structures often utilized unique and not obvious control strategies to solve the control problems confronting the animat. The section of the control structure that was active was determined by a combination of evolving bottom-up reactive drives and top-down goal seeking drives. As a bottom-up reactive/opportunistic drive was triggered, it was fulfilled by a cascade of top-down invoked skills. The nested Q-learning algorithm effectively partitioned the control problem into many small evaluations functions to be combined and recombined into different solutions rather than having solutions locked into a single monolithic evaluation function.

References

[1] Barto, A.G., Sutton, R.S. and Watkins C.H. (1989), Learning and Sequential Decision Making, *COINS Technical Report.*

[2] Digney B. L. (1994), A Distributed Adaptive Control System for a Quadruped Mobile Robot, *From animals to animats 3: SAB 94*, Brighton, UK, August 1994, pp. 344-354, MIT Press-Bradford Books, Massachussets.

[3] Albus, J.S. (1991), Outline of a Theory of Intelligence. *IEEE Transactions on Systems, Man and Cybernetics*, 21, 3, pp. 473-509.

[4] Digney, B.L. (1996), Shaping Emergent Control Structures with Scaffolding Actions and Staged Learning, *Artificial Life V*, May, 1996, Nara-Ken, Japan. (to appear)

[5] Tyrrel, T., (1992), The Use of Hierarchies for Action Selection, *From animals to animats 2: SAB 92*, pp. 138-148, MIT Press-Bradford Books, Massachussets.

[6] Meyer, J.A. and Guillot, A. (1994), From SAB90 to SAB94, *From animals to animats 3: SAB 94*, Brighton UK, August 1994, pp. 2-11, MIT Press-Bradford Books, Massachussets.

[7] Maes, P. and Brooks, R.A., (1990), Learning to coordinate behaviors, *Eighth National Conference on Artificial Intelligence* pp. 796-802, 1990.

[8] Mahadevan, S. and Connell, J. (1991), Automatic programming of behavior based robots using reinforcement learning, *Nineth National Conference on Artificial Intelligence*, Anaheim, CA, 1991.

[9] Long-Ji, L. (1993), Hierarchical learning of robot skills, *IEEE International Conference on Neural Networks*, San Francisco, 1993, pp. 181-186.

[10] Dayan, P. and Hinton, G.E., Feudal reinforcement learning, *Advances in Neural Information Processing Systems 5*, San Mateco, CA, Morgan Kaufman, 1993.

[11] Singh, P.S., (1992), Transfer of learning by composing solutions of elemental sequential tasks, *Machine Learning, 8(3/4)*, pp. 323-339, 1992.

[12] Kaelbling, L.P., (1993), Hierarchical learning in stochastic domains: Preliminary results, *Tenth International Conference on Machine Learning*, Amherst, MA, 1993.

[13] Thurn, S.B., (1992), *Efficient exploration in reinforcement learning*, Technical Report CMU-CS-92-102, Carnegie Mellon University, School of Computer Science, 1992.

A model of operant conditioning for adaptive obstacle avoidance

Paolo Gaudiano†, Eduardo Zalama‡, Carolina Chang†, and Juan López Coronado‡
† Boston University, Department of Cognitive and Neural Systems
677 Beacon Street, Boston, MA 02215, USA

‡ Department of Control Systems Engineering, University of Valladolid
Paseo del Cauce s/n, 47011, Valladolid, SPAIN
E-mail: gaudiano@bu.edu, eduzal@dali.eis.uva.es, cchang@bu.edu, coronado@cpd.uva.es

May 1, 1996

Abstract

We have recently introduced a self-organizing adaptive neural controller that learns to control movements of a wheeled mobile robot toward stationary or moving targets, even when the robot's kinematics are unknown, or when they change unexpectedly during operation. The model has been shown to outperform other traditional controllers, especially in noisy environments. This article describes a neural network module for obstacle avoidance that complements our previous work. The obstacle avoidance module is based on a model of classical and operant conditioning first proposed by Grossberg (1971). This module learns the patterns of ultrasonic sensor activation that predict collisions as the robot navigates in an unknown cluttered environment. Along with our original low-level controller, this work illustrates the potential of applying biologically inspired neural networks to the areas of adaptive robotics and control.

1 Introduction

Traditional autonomous robots are controlled by either direct program control, teaching pendants, model reference, or conventional adaptive control techniques. Recently, neural networks have emerged as a promising solution to the problem of adaptive control in uncertain environments (Katayama & Kawato, 1991; Kawato, Maeda, Uno, & Suzuki, 1990; Jordan & R.A., 1990). This approach has made it possible not only to develop experimental robots but also to understand human and animal movement control. Much of the work to date has focused on dynamic motor control and sensory-motor coordination, with recent successful applications in robotics (Kawato, Furukawa, & Suzuki, 1987; Kuperstein, 1991). In the realm of robot control, several authors have utilized neural networks to train robots to reach targets or follow prescribed trajectories (Baloch & Waxman, 1991; Nagata, Sekiguchi, & Asakawa, 1990; Nguyen & Widrow, 1990; Plumer, 1992; Pomerleau, 1993).

Neural networks have been shown useful especially for their ability to learn complex nonlinear mappings that can capture the kinematics or dynamics of a robot or other plant. However, little advantage has been taken of some of the more biologically inspired neural networks that have been used to understand how biological organisms survive autonomously in a constantly changing environment. One of the aspects to consider when an animal or intelligent machine has to operate in an unknown environment is that it must learn to predict the consequences of its own actions. By learning the causality of environmental events, it becomes possible to predict future and new events.

We have recently introduced a neural network model for the low-level control of a mobile robot (Zalama, Gaudiano, & López-Coronado, 1995; Gaudiano, Zalama, & López Coronado, 1996). The model, known as NETMORC, learns to control a differential drive robot with no knowledge of the robot's odometry or kinematics, through a cycle of learning-by-doing. In this article the NETMORC model is extended to include a module for adaptive avoidance behavior. The obstacle avoidance module is based on a neural network model of classical and operant conditioning (Grossberg, 1971; Grossberg & Levine, 1987). After a brief description of operant conditioning, we summarize Grossberg's original model, and describe its application to obstacle avoidance using our NETMORC scheme.

2 The NETMORC architecture

The NETMORC model is able to control the navigation of a mobile robot in a nonstationary environment by learning in an unsupervised manner the inverse and forward kinematics of a differential-drive robot. Movement is performed by selecting the angular velocities for the

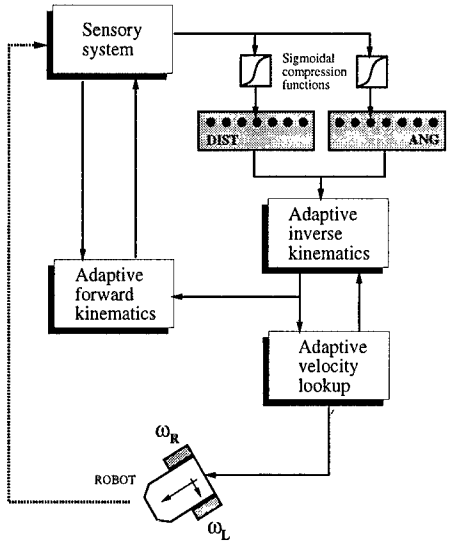


Figure 1: The NETMORC architecture learns the inverse kinematics of the robot in order to guide it to a goal. It also includes an internal feedback representing the forward kinematic learning of the mobile robot. This allows the robot to perform "blind reaching" when the sensory feedback is not working.

motors attached to the propulsive wheels of the robot. The basic scheme of the NETMORC architecture is represented in figure 1.

Following an initial learning phase, the controller architecture allows movement between arbitrary points through sensor (e.g., visual or ultrasonic) information. It is important to note that rather than learning explicit trajectories, the controller learns the relationship between angular velocities and the magnitude and direction of the resulting movement. This approach solves the inverse kinematic problem, so that sensory information in spatial coordinates can generate the appropriate wheel angular velocities to move the mobile robot to a desired goal. An endogenous random generator (ERG) (Gaudiano & Grossberg, 1991) produces random angular velocities w_L and w_R for the two wheels, leading to random movements of the robot. At regular time increments, the sensory system determines the robot's displacement resulting from the momentary values of w_L and w_R. These displacements are coded in the populations of neurons labeled DIST and ANG in Fig. 1. The robot learns the inverse mapping through associations between wheel velocities and linear and angular displacements. In this way the robot learns the angular velocities required to generate a specific movement.

After learning has occurred, the neural controller allows movement between arbitrary points through sensory information. The sensory system computes the distance and angle to target from an initial point. These are coded by each node in the DIST and ANG populations through a sigmoidal compression function (so as to allocate more nodes for small distance and angle values). Then the angular velocity of each wheel's motor is generated through the adaptive inverse kinematics and the adaptive velocity lookup modules as the robot approaches the target. In a recent publication (Gaudiano et al., 1996) we were able to demonstrate that this scheme is Lyapunov stable, guaranteeing that the robot will move to the desired target after sufficient training.

It should be noted that although for clarity of explanation we refer to distinct learning and performance phases, the learning can take place continuously even during normal reaching behavior. This enables the neural controller to adapt to changes in the robot's plant, such as wheel wear and other miscalibrations, while moving to a target.

Moreover, the forward odometry of the mobile robot is also learned. This allows the robot to move even if the sensory system stops working, or operates slower than needed. The internal feedback updates the current position of the robot through the adaptive forward odometry module. A complete description of each adaptive module can be found in Zalama et al. (1995), Gaudiano et al. (1996).

One property of this neural controller is that the robot trajectory is automatically carried out as the wheel velocities are generated. Thus, there is no need for a previous phase of path planning. However, the controller obviously has no knowledge of obstacles or how to avoid them. Nonetheless, the modular nature of the NETMORC architecture makes it easy to add modules that can steer the robot around obstacles toward its target. This can be accomplished simply by interposing a module between the sensory system and the robot's DIST and ANG populations. Specifically, obstacle avoidance modules can continuously generate a "fictitious" target to navigate around the obstacles, and feed the DIST and ANG information to the NETMORC, which will follow this fictitious target.

The remainder of this article describes an obstacle avoidance scheme based on a neural network model of classical and operant conditioning first proposed by Grossberg (Grossberg, 1971, 1986; Grossberg & Levine, 1987). The model learns the relationship between patterns of sensor activities and the effects of the robot's actions. Specifically, the robot learns the patterns of ultrasonic sensor activations that predict an impending collision as the robot is navigating in an unknown cluttered environment. After sufficient cycles of learning-by-doing, the robot is able to avoid obstacles reliably in arbitrary configurations.

3 Adaptive obstacle avoidance through operant conditioning

One of the aspects to consider when an animal or intelligent machine has to operate in an unknown environment is that it must learn to predict the consequences of its own actions. By learning the causality of environmental events, it becomes possible to predict future and new events. Models of classical and operant conditioning have emerged from the field of psychology in order to try to explain how an organism can achieve autonomous behavior in a constantly changing environment (Grossberg, 1971; Rescorla & Wagner, 1972; Grossberg, 1982; Sutton & Barto, 1981, 1990).

In the classical conditioning paradigm, learning occurs by repeated association of a Conditioned Stimulus (CS), which normally has no particular significance for an animal, with and Unconditioned Stimulus (UCS), which has significance for an animal and always gives rise to an Unconditioned Response (UCR). For example, a dog that repeatedly hears a bell (CS) before being fed (UCS) will eventually begin to salivate (UCR) when the bell is heard. The response that comes to be elicited by the CS after classical conditioning is known as the Conditioned Response (CR).

Another related form of learning is known as *operant conditioning*. In this case an animal learns the consequences of its actions. More specifically, the animal learns to exhibit more frequently a behavior that has led to a reward, and to exhibit less frequently a behavior that has led to punishment. For example, a hungry cat placed in a cage from which it can see some food will learn to press a lever that allows it to escape the cage to reach the food. In this situation, the animal cannot simply wait for things to happen, but it must generate different behaviors and to learn which are effective. This kind of learning has also been referred to as *reinforcement learning* (Sutton & Barto, 1981). The main problem in modeling this form of learning is how to learn which of a large array of behaviors has produced the reward.

4 Grossberg's conditioning model

In 1971, Grossberg proposed a model of classical and operant conditioning, which was designed to account for a variety of behavioral data. The model was refined in several subsequent publications. In 1987, Grossberg & Levine published the results of computer simulations of the main components of the conditioning circuit. Grossberg & Levine's (1987) implementation of Grossberg's conditioning circuit is shown in figure 2. This model was used to explain a number of phenomena from classical conditioning.

In the Grossberg & Levine model the *sensory cues* (both CSs and UCS) are stored in Short Term Memory (STM) within the population labeled S, which includes

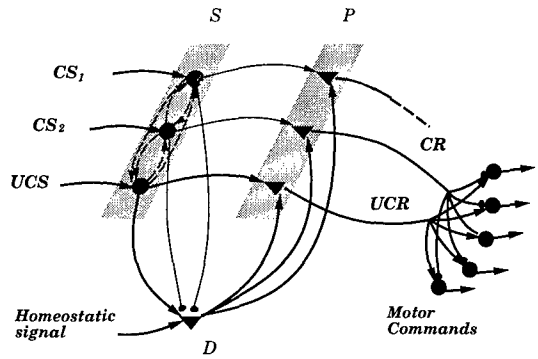


Figure 2: Conditioning circuit proposed by Grossberg & Levine. See text for details.

competitive interactions to ensure that the most salient cues are contrast-enhanced and stored in STM while less salient cues are suppressed. At the bottom of the figure the *drive node* D is represented. The drive node represents information about the organism's internal state, and conditioning can only occur when the drive node is active. For example, and animal that sees food will not be motivated to eat unless it is hungry. Thus the joint combination of an internal homeostatic hunger signal and an external food stimulus are required to generate the eating behavior. To represent the need for joint external and internal signals, the drive node is represented as a triangular node, symbolizing a *polyvalent* cells, i.e., one that requires the convergence of two inputs in order to become active. The cells in the population labeled P are also polyvalent for reasons that will be described below. Finally, the neurons at the far right of the figure represent the response (conditioned or unconditioned), and are thus presumably connected to the motor system.

The design of figure 2 satisfies a number of fundamental constraints. Most important are the requirements that: (1) initially, only the UCS must be able to cause a response (the UCR); (2) after conditioning, the CS must be able to elicit a response similar to the UCR; (3) in order for learning to take place, the CS and UCS must be presented nearby in time.

The system satisfies these requirements as follows: initially it is presumed that only the UCS has a strong connection to the drive node D. When the UCS is turned on (e.g., a shock), it activates the drive node, which in turn sends activation up toward the polyvalent cells P. The P cell that receives joint input from the UCS and D nodes becomes active, and "reads out" the UCR on the motor cells. When a CS (e.g., a light) is presented by itself prior to conditioning, it cannot activate the D node, and thus it cannot activate its P cell, and no action is generated. However, if the UCS is turned on shortly after the CS has become active, then the D node becomes active, and the connection from the CS to the D node increases through a simple associative learning rule. At

the same time, the D node will also briefly activate *all* P cells that are receiving sensory input, and thus the P cell corresponding to the CS will be active, and it will learn the pattern of motor activity being generated by the UCS. If the pairing is repeated several times, the C-S will develop strong connections to the D node, while its polyvalent (P) cell will learn to "imitate" the UCR. Eventually the CS connection to the drive node becomes strong enough that activation of the CS leads to activation of the drive node D even in the absence of a UCS. The joint activation of the D node and the CS node leads to activation of the polyvalent cell P, which then reads out a response similar to the UCR, that is, the CR.

Notice that as mentioned above the drive node must receive both a sensory signal (such as the sight of food) and a homeostatic signal (such as hunger) in order to be activated. However, when the CS and UCS represent a fearful cue, it is assumed that the homeostatic signal corresponds to some form of "survival drive," which is presumably always active in normal animals. In this case, as we do below, one can simply assume that the drive node only needs a strong sensory cue to become active.

An important characteristic of the model is its dynamical nature and temporal causality. Temporal causality refers to the fact that stimulus and response can become associated even though they are presented at different times in different trials. For example, when an animal presses a lever to get food, it will learn the effect of its action regardless of exactly how quickly the food is presented (within a window of several seconds).

The Grossberg & Levine model is able to reproduce other paradigms from the study of learning, such as blocking and second-order conditioning (Grossberg & Levine, 1987). Second-order conditioning is useful as it allows a CS previously paired with a UCS, to act as a UCS for other conditional stimuli. For instance, a bell repeatedly paired with shock eventually comes to elicit fear. If a light is now repeatedly presented shortly before onset of the bell, even though the shock is never turned on, the light will also come to elicit fear. This form of "higher order" conditioning is extremely useful as it allows animals to learn early predictors of important events.

5 Conditioning and obstacle avoidance

In this section we describe how we have applied Grossberg's conditioning model to the problem of obstacle avoidance with a mobile robot. The mobile robot we have used in the simulations is a differential-drive robot, as shown in figure 3 equipped with an array of range sensors.

We have previously introduced a neural network controller for this type of mobile robot, which learns the robot's forward and inverse odometry through a learning-by-doing cycle (Zalama et al., 1995; Gaudiano et al., 1996). That model, which was described briefly in an earlier section, includes two neural populations that code the distance and angle to a target as registered by the robot's sensory system. The distance and angle information is used to generate the wheel velocities required to move the robot toward the target.

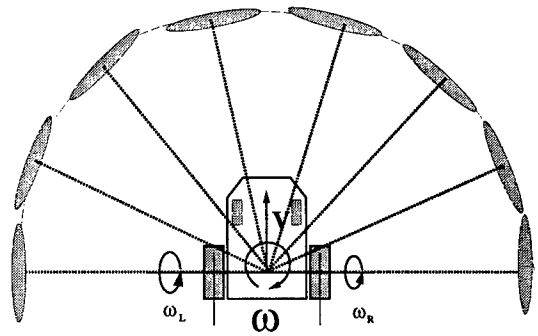


Figure 3: Mobile robot structure. The mobile robot has a set of 8 ultrasonic sensors uniform and frontal distributed through the robot.

In the present work we combine the conditioning model of figure 2 with the NETMORC in such a way that the robot can learn to avoid obstacles by modifying its angular velocity when it encounters obstacles in its path. Note that nodes in the ANG population are directly proportional to the angular velocity that will be generated by NETMORC. Hence for clarity in the remainder of the paper we frequently refer to angular velocity nodes or angle nodes interchangeably.

Figure 4 illustrates the scheme we used to represent angular velocities. In this figure the leftmost node represents an angular velocity of $-w_m rad/s$, the right node represents an angular velocity of $w_m rad/s$ (where w_m is the maximum angular velocity developed by the robot), and the central node corresponds to a straight line movement. The activation pattern over the population is used to determine the wheel velocities that will move the robot. The map includes a sigmoidal transformation, whereby angular velocities close to zero are represented by a greater number of nodes for finer control. The sigmoidal function that selects the most active node in the map as a function of the angular velocity is given by:

$$n_d = \begin{cases} \frac{N}{2} + \frac{N(a_w+0.5w_m)w}{w_m(a_w+w)} & \text{if } w > 0 \\ \frac{N}{2} + \frac{N(a_w+0.5w_m)w}{w_m(a_w-w)} & \text{otherwise} \end{cases} \quad (1)$$

where n_d represents the most active node, w is the angular velocity of the robot, N is the number of nodes in the map, w_m is maximum angular velocity, and a_w controls the steepness of the sigmoid.

In our simulations, instead of only activating a sin-

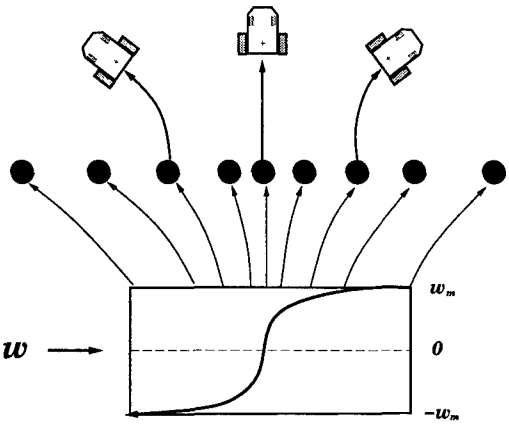


Figure 4: Angular velocity map. Each spatial position represents an angular velocity developed by the robot. The transformation has been performed by means of a sigmoidal function, which yields a higher density of nodes for velocity values close to zero.

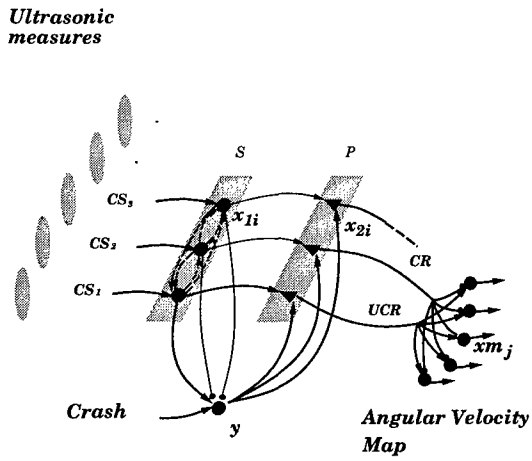


Figure 5: Conditioning model for obstacle avoidance. The ultrasonic range data represents the conditioned stimuli; the crash is the unconditioned stimulus. After conditioning, the pattern of activity across the ultrasonic sensors can predict a collision and change the angular velocity to avoid the obstacle.

gle node in the population, we activate a neighborhood of nodes, with the position of maximal activation corresponding to the preferred angular velocity or direction of movement, and the activation of nearby nodes falling off as a Gaussian. We will show later that this form of distributed activity lends itself well to the problem of obstacle avoidance.

The simulated mobile robot has eight ultrasonic sensors angularly distributed every 25.5^o, covering the frontal plane of the robot as figure 3 shows. In the simulations we have assumed a maximum range of $5m$ for each sensor, and that a collision occurs when any of the sensors measures a distance under $1m$. An alternative solution in a practical situation could utilize information from bump sensors positioned on along the robot's perimeter. However, it is important to point out that the conditioning model has no knowledge of the robot's geometry or of the location of the sensors, as we will show below.

In figure 5 the proposed model for obstacle avoidance is shown. In this case each sensory cue (or CS) corresponds to the *complement* of the signal from an ultrasonic sensor, that is, the raw reading from the sensor subtracted from the maximum value of each ultrasonic sensor, so that closer obstacles are represented by larger activity. The unconditioned stimulus (UCS) in this case corresponds to a collision detected by the robot. For simplicity, rather than treating the collision detector as a UCS with a strong, fixed connection to the drive node, we let the collision signal activate directly the drive node. This corresponds to the situation discussed above in which an internal "survival" signal is always impinging upon the drive node. This simplification is reasonable as long as a single kind of UCS and drive are necessary.

The overall idea behind the model in figure 5 is that whenever the robot collides with an obstacle, learning in the circuit will create a connection between the current pattern of ultrasonic sensor activity and the angular velocity that the robot had at the time of collision. Later activation of a similar sensor pattern will cause inhibitory activation of those angular velocities that would have caused a crash, causing the robot to change direction and steer away from an obstacle.

One of the main properties of the model is its real-time function, in the sense that it is not necessary to separate explicitly learning from normal operation. However, in order to achieve faster learning we have performed an initial learning phase where the robot performs random exploratory movements in a cluttered environment, activating sequentially each node from the angular velocity map, and thus sampling various patterns of activity that lead to collision.

As the robot travels, it takes measures from the ultrasonic sensors. The sensor activation values (actually their complement, as described above) are contrast-enhanced and stored within the STM field S. The population S, which was originally modeled by Grossberg as a *recurrent competitive field*, removes the inherent noise from the ultrasonic sensors and enhances the activity of those sensors receiving maximal activation, that is, those registering the closest obstacles. A more complete description of the properties of this kind of network can be found elsewhere (Grossberg, 1973, 1982). We have used a simplified discrete-time version of the recurrent competitive field, which quickly and efficiently normalizes and contrast-enhances the sensor activations. Specifically, the activation x_{1i} of each neuron in population S is given by

$$x_{1i}(t) = M_x \frac{[R - I_i(t)]^+}{\sum_j x_{1j}(t-1)} - (1 - M_x)x_{1i}(t-1) \quad (2)$$

where I_i is the "raw" ultrasonic measures, R is the maximum range for each ultrasound, M_x is a constant that determines how much the previous activation is weighted relative to the current input, and the notation $[x]^+$ represents half-wave rectification (returns only those values of $x > 0$). The summation is taken over all ultrasonic activations, thus ensuring normalization over the entire population S.

Prior to conditioning, the drive node D can only be activated when the robot collides against obstacles. After conditioning, sufficient activation of a pattern of sensors can also activate the drive node. This permits higher-order conditioning, so that after the initial learning stages the ultrasonic activations can themselves predict the collision, and lead to conditioning of other sensor patterns. The activation y of the drive node is given by:

$$y(t) = \sum_i x_{1i}(t)z_{1i}(t) - T_y + U_{CS}(t) \quad (3)$$

where U_{CS} represents the collision of the robot ($U_{CS} = 1$ if collision just occurred, and $U_{CS} = 0$ otherwise), z_{1i} is the adaptive weight connecting the sensory node x_{1i} to the drive node, and T_y is a threshold that controls how easily the drive node is activated, and thus indirectly controls how easily higher-order conditioning will occur.

The activation x_{2i} of polyvalent cells is given by:

$$x_{2i}(t) = x_{1i}(t)f(y(t)) \quad (4)$$

where $f(y(t))$ is given by:

$$f(y(t)) = \begin{cases} 1 & \text{if } y(t) > 0 \\ 0 & \text{otherwise} \end{cases} \quad (5)$$

Two different kinds of learning take place: the learning that couples sensory nodes (ultrasounds) with the drive node (the collision), and the learning of the angular velocity pattern that existed just before the collision. The first type of learning follows an associative learning law with decay:

$$z_{1i}(t) = Lz_{1i}(t-1) + Px_{1i}(t)f(y(t)) \quad (6)$$

where P is the learning rate, L is the weight decay.

The equation that learns the velocity map is also of an associative form:

$$zm_{i,J}(t) = zm_{i,J}(t-1) - Mx_{2i}(t)[xm_J(t-1) + zm_{i,J}(t-1)] \quad (7)$$

where $zm_{i,J}$ represents the adaptive weights from the polyvalent cells to the most active (winning) node J within the angular velocity map, M is the learning rate. However, in this case the learned weights are negative (the steady-state solution is reached when $xm_J + zm_{i,J} = 0$), thus when the robot collides against obstacles, the above rule learns to inhibit the actual direction of movement. Note that this learning rule is equivalent to learning a pattern of activity over a map of neurons that are coupled through mutual inhibitory connections with the angular velocity map. The use of negative connection weights is computationally more parsimonious.

Once learning has occurred, the activation of the angular velocity map is given by two components. A first excitatory component, which is generated directly by the sensory system, reflects the angular velocity required to reach the target without the influence of obstacles. We have shown previously how this signal could be derived from the sensors (Gaudiano et al., 1996); for simplicity here we assume that the angular velocity is proportional to the angle between the robot's heading and the target. The second, inhibitory component, generated by the conditioning model in response to sensed obstacles, moves the robot away from the obstacles as a result of the activation of ultrasound signals in the conditioning circuit. The equation that reflects this behavior is given by:

$$xm_j(t) = \exp\left[-(j - n_d(t))^2/\sigma^2\right] + \sum_i x_{2i}(t)zm_{i,j}(t) \quad (8)$$

where $n_d(t)$ is the index of the node that represents the desired angular velocity to reach the target without obstacles, and σ is the standard deviation of the Gaussian.

The reason for using a Gaussian distribution of activity is depicted in figure 6. When an excitatory Gaussian is combined with an inhibitory Gaussian at a slightly shifted position, the resulting net pattern of activity exhibits a maximum peak that is shifted in a direction *away* from the peak of the inhibitory Gaussian. Hence the presence of an obstacle to the right causes the robot to shift to the left, and *vice versa*.

In our previous work the NETMORC architecture was used to learn autonomously the relationship between distance and angle to target and the corresponding wheel angular velocities. In the present example for simplicity we use the kinematic equations directly to move the simulated robot (this is simply a computational expedient that does not necessitate learning of the robot's odometry). Specifically, for a given pattern of activity of the angular velocity nodes, we select an angular velocity according to:

$$w = \begin{cases} \frac{w_m a_w [J(t) - N/2]}{N(a_w + 0.5w_m) - w_m[J(t) - N/2]} & \text{if } J(t) > N/2 \\ \frac{w_m a_w [J(t) - N/2]}{N(a_w + 0.5w_m) + w_m[J(t) - N/2]} & \text{otherwise} \end{cases} \quad (9)$$

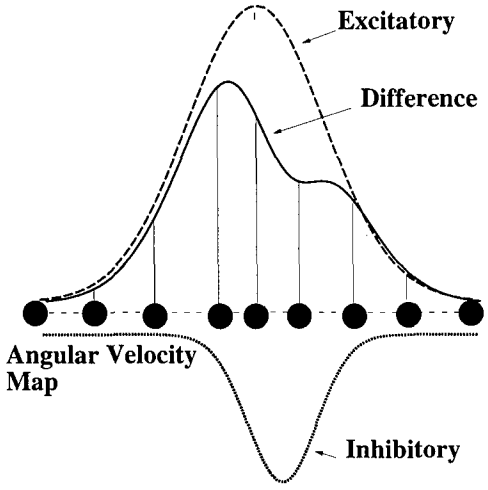


Figure 6: Positive Gaussian distribution represents the angular velocity without obstacles, and negative distribution represents activation from the conditioning circuit. The difference represents the angular velocity that will be used to drive the robot. Notice how the maximum of the excitatory Gaussian is shifted by the inhibitory Gaussian.

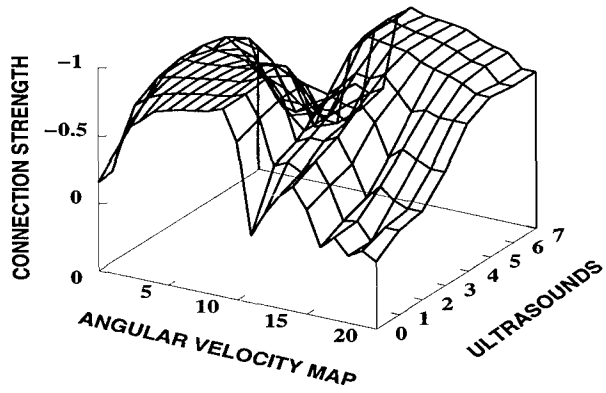


Figure 7: Representation of the adaptive connections $x_2 \rightarrow xm$. This map intrinsically codes the location of the ultrasonic sensors, in such way that when echoes are received in a given direction, that direction is inhibited in the angular velocity map.

This function is essentially the inverse of the sigmoid described by equation 1 above, where $J(t) = max_j[xm_j(t)]$ represents the node number with the largest activity in the angular velocity map.

6 Experimental results

In this section we show our preliminary results on the model's performance. In a first stage, we let the model develop a set of weights by letting the robot perform movements at different angular velocities in an environment cluttered with obstacles, by activating sequentially the nodes in the angular velocity map. After this initial learning phase the robot is able to avoid obstacles in arbitrary positions.

More specifically, each obstacle is defined as a solid polygon, and the distance between each sensor and all obstacles within the sensor range is calculated algorithmically, and used to generate the sensor activations as described above. The position and size of the targets is chosen randomly, and the robot is systematically driven to move left, straight, and right several times so as to learn a variety of sensor patterns that predict collisions (defined in this simple case when any one of the sensors gives a reading below one meter).

Figure 7 shows an example of the projections of the adaptive connections developed between the sensory nodes x_2 and the angular velocity map xm. In the figure one can see for example how angular velocities that make the robot turn to the right (nodes close to 20 in the figure) are inhibited when the robot receives ultrasonic sensor signals to the right (values close to 7). The weight map is not perfectly symmetric because of the random presentation of obstacles during training. Using a longer training phase and a slower learning rate should reduce this asymmetry.

Figure 8 illustrates the robot's performance in the presence of several obstacles. The robot starts from the initial position labeled 1 and reaches a sequence of points 2-7 along the path shown by the dashed line. During the movements, whenever the robot is approaching an obstacle, the inhibitory profile from the conditioning circuit changes the selected angular velocity and makes the robot turn away from the obstacle. The presence of multiple obstacles at different positions in the robot's sensory field causes a complex pattern of activation that steers the robot between obstacles.

The learning rule includes a passive decay term which can in principle cause weight decay if the robot never experiences a collision during normal performance. This was not a problem for the relatively brief simulations reported here, but it would have to be addressed in longer simulations or real situations. The problem, however, can be easily addressed in two ways. The simplest way is simply to "freeze" the weights after some initial training period. This is a traditional technique used in neural network approaches, which is satisfactory if no changes take place in the sensor configuration or other aspects of the robot. If changes do occur (for instance, addition of other sensors), the robot can be re-trained. A second solution is to utilize a more complex version of Grossberg's conditioning circuit (Grossberg & Schmajuk, 1987), one that prevents sensory cues from losing their strength unless a given sensory cue (CS) that was previously paired with a collision signal, is actively paired with a signal representing the absence of collision. This technique, known as "extinction" in the psychological literature,

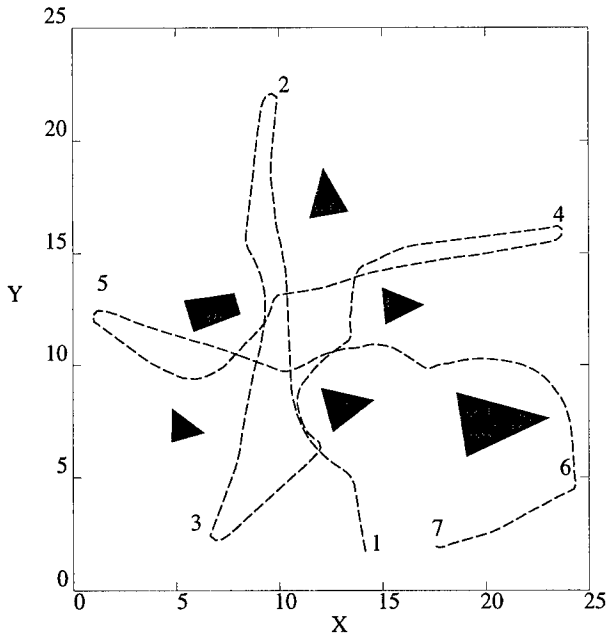


Figure 8: Trajectory followed by the robot in presence of six obstacles (shaded polygons). The robot travels from point 1 to point 7. Distances are expressed in meters. Parameters employed in simulations: $N = 21$, $w_m = 2.0$, $a_w = 0.3$, $M_x = 0.6$, $R = 5.0$, $T_y = 0.2$, $L = 0.9$, $P = 0.1$, $M = 0.05$, $\sigma = 80$.

was demonstrated to work by Grossberg and Schmajuk (1987); however, we felt that the increased complexity was not appropriate for this initial study, though we plan on including it in our future work.

7 Conclusions

In this article we have described preliminary results with a model of operant conditioning that is used to learn obstacle avoidance for a wheeled mobile robot equipped with an array of ultrasonic sensors. The robot progressively learns to avoid the obstacles without the necessity of external supervision, but simply through the "punishment" signals produced by the collision of the robot. One of the main properties of the model is that it is not necessary to know the robot's geometry nor the configuration of ultrasonic sensors on the robot's surface, because the robot learns from past experiences to avoid directions of movement that make the robot collide against the obstacles.

We are extending this models of conditioning to develop more complex behaviors. In particular, we are investigating conditioning circuits that permit the robot to choose among different behaviors (avoid, escape, wall following, etc.) depending on the moment-by-moment combination of sensory information and its internal necessities. For example, a more complex system of sensory and drive nodes could be used to modulate how much the robot will try to avoid obstacles depending on its necessity to recharge its batteries. One of our short-term goals is to implement the present model into a real robot platform, as we have done in our previous work.

8 Acknowledgments

We wish to thank the neural networks group of the Systems and Control Engineering Department of the ETSII of the University of Valladolid for their valuable contribution to the realization of this work. P. Gaudiano is supported by ONR N00014-95-1-0409, and by an ONR Young Investigator Award (ONR N00014-96-1-0772).

Reference

Baloch, A., & Waxman, A. (1991). Visual learning, adaptive expectations, and behavioralconditioning of the mobile robot MAVIN. *Neural Networks, 4*, 271–302.

Gaudiano, P., & Grossberg, S. (1991). Vector associative maps: Unsupervised real-time error-based learning and control of movement trajectories. *Neural Networks, 4*, 147–183.

Gaudiano, P., Zalama, E., & López Coronado, J. (1996). An unsupervised neural network for real-time, low-level control of a mobile robot: noise resistance, stability, and hardware implementation. *IEEE SMC*. In press. Preliminary version appears as *Boston University Technical Report*, **CAS/CNS-TR-94-019**.

Grossberg, S. (1971). On the dynamics of operant conditioning. *Journal of Theoretical Biology, 33*, 225–255.

Grossberg, S. (1982). A psychophysiological theory of reinforcement, drive, motivation and attention. *Journal of Theoretical Neurobiology, 1*, 286–369.

Grossberg, S., & Levine, D. (1987). Neural dynamics of attentionally modulated Pavlovian conditioning: blocking, interstimulus interval, and secondary reinforcement. *Applied Optics, 26*, 5015–5030.

Grossberg, S., & Schmajuk, N. A. (1987). A neural network architecture for attentionally-modulated Pavlovian conditioning: Conditioned reinforcement, inhibition, and opponent processing. *Psychobiology, 15*, 195–240.

Grossberg, S. (1973). Contour enhancement, short-term memory, and constancies in reverberating neural networks. *Studies in Applied Mathematics, 52*(3), 217–257.

Grossberg, S. (Ed.). (1982). *Studies of Mind and Brain: neural principles of learning, perception, development, cognition and motor control.* Reidel, Boston.

Grossberg, S. (Ed.). (1986). *The Adaptive Brain I: Cognition, Learning, Reinforcement, and Rhythm.* Elsevier/North-Holland, Amsterdam.

Jordan, M. A., & R.A., J. (1990). Learning to control an unstable system with forward modelling. In Touretzky (Ed.), *Advances in Neural Information Processing System 2, D.S.*, pp. 324–331. Morgan Kaufmann.

Katayama, M., & Kawato, M. (1991). Learning trajectory and force control of an artificial muscle arm by parallel-hierarchical neural network model. In Lippman, R. e. a. (Ed.), *Adavances in Neural Information Processing Systems 3*, pp. 436–442. Morgan Kaufmann.

Kawato, M., Furukawa, K., & Suzuki, R. (1987). A hierarchical network model for control and learning of voluntary movement. *Biological Cybernetics, 57*(2), 169–185.

Kawato, M., Maeda, Y., Uno, Y., & Suzuki, R. (1990). Trajectory formation of arm movement by cascade neural network model based on minimum torque-change criterion. *Biological Cybernetics, 62*, 275–288.

Kuperstein, M. (1991). INFANT neural controller for adaptive sensory-motor coordination. *Neural Networks, 4*, 131–145.

Nagata, S., Sekiguchi, M., & Asakawa, K. (1990). Mobile robot control by a structured hierarchical neural network. *IEEE Control Systems Magazine, 10*, 69–76.

Nguyen, D., & Widrow, B. (1990). Neural networks for self-learning control systems. *IEEE Control Systems Magazine, 10*(3), 18–23.

Plumer, E. (1992). Neural network structure for navigation using potential fields. In *International Joint Conference on Neural Networks*, Vol. I, pp. 327–332 Baltimore, Maryland.

Pomerleau, D. A. (1993). Knowledge-based training of artificial neural networks for autonomous robot driving. In *Springer Series in Information Science.* Kluwer Academic Publishers, Boston, Dordrenht, London.

Rescorla, R. A., & Wagner, A. R. (1972). A theory of pavlovian conditioning: variations in the effectiveness of reinforcement and nonreinforcement. In Black, A. H., & Prokasy, W. F. (Eds.), *Classical Conditioning II*, chap. 3, pp. 64–99. Appleton, New York.

Sutton, R. S., & Barto, A. G. (1981). Toward a modern theory of adaptive networks: Expectation and prediction. *Psychological Review, 88*, 135–170.

Sutton, R. S., & Barto, A. G. (1990). Time-derivative models of Pavlovian reinforcement. In Moore, J. W., & Gabriel, M. (Eds.), *Learning and Computational Neuroscience.* MIT Press.

Zalama, E., Gaudiano, P., & López-Coronado, J. (1995). A real-time, unsupervised neural network for the low-level control of a mobile robot in a nonstationary environment. *Neural Networks, 8*(1), 103–123.

Learning Navigational Behaviors using a Predictive Sparse Distributed Memory

Rajesh P.N. Rao and Olac Fuentes
Department of Computer Science
University of Rochester
Rochester, NY 14627
{rao,fuentes}@cs.rochester.edu

Abstract

We describe a general framework for the acquisition of perception-based navigational behaviors in autonomous mobile robots. A self-organizing sparse distributed memory equivalent to a three-layered neural network is used to learn the desired transfer function mapping sensory input into motor commands. The memory is initially trained by teleoperating the robot on a small number of paths within a given domain of interest. During training, the vectors in the sensory space as well as the motor space are continually adapted using a form of competitive learning to yield basis vectors aimed at efficiently spanning the sensorimotor space. After training, the robot navigates from arbitrary locations to a desired goal location using motor output vectors computed by a saliency-based weighted averaging scheme. The pervasive problem of perceptual aliasing in non-Markov environments is handled by allowing both current as well as the set of immediately preceding perceptual inputs to predict the motor output vector for the current time instant. Simulation results obtained for a mobile robot, equipped with simple photoreceptors and infrared receivers, navigating within an enclosed obstacle-ridden arena indicate that the method performs successfully in a variety of navigational tasks, some of which exhibit substantial perceptual aliasing.

1 Introduction

Behaviors that realize goal-directed, collision-free navigation in unstructured environments form an extremely important component of the behavioral repertoire of any autonomous mobile robot. Traditional sensorimotor controllers relied on fixed robot behaviors that were prewired by the control designer. Such an approach suffered from at least two problems: (a) the increasing complexity of the design process when scaling up from toy-problems to real environments and (b) the inherent inflexibility of prewired behaviors due to their inability to adapt to circumstances unforeseen at design time. The first problem is addressed by Brooks [4], who proposes a *hierarchical behavior-based decomposition* of control architectures. Such behavior-based robot architectures have been shown to accomplish a wide variety of tasks in an efficient manner [7, 19]. The second problem has been addressed by endowing robots with the ability to autonomously learn behaviors either from experimentation and dynamic interaction with their environment or via teleoperation. A number of learning algorithms have been used for this purpose including neural networks [29, 30, 37], evolutionary algorithms/genetic programming [3, 6, 18], reinforcement learning [2, 13, 20], hill-climbing routines [10, 27], and self-organizing maps [16, 24, 35].

In the context of robot navigation, a popular approach has been the construction and use of global maps of the environment [9]. Such an approach, while being intuitively appealing, faces the difficult problem of robot localization *i.e.* establishing reliable correspondences between noisy sensor readings and geometrical map information in order to estimate current map position. In addition, the global map inherits many of the undesirable properties of centralized representations that Brooks [4] identifies in traditional robot controllers. An alternative behavior-based approach to navigation is proposed by Mataric [22] (see also [8, 17]). This method avoids the problems involved in creating and maintaining global representations by utilizing only local information as provided by landmarks along a navigational path. A potential problem with such landmark-based approaches is that incorrect landmark matches or changes in the landmarks themselves might result in catastrophic failure of the navigational system.

In this paper, we propose a robust behavior-based approach to goal-directed mobile robot navigation based on a predictive sparse distributed memory. A "teaching-by-showing" paradigm is adopted to initially train the memory for achieving a prespecified navigational behavior; such an approach drastically cuts down on the number of training trials needed as compared to learning by trial and error or by random exploration of the environment. In our case, the robot is teleoperated (via a remote joystick) on a small number of training paths

within the given environment. During training, the vectors in the perception space as well as the motor space are adapted using a form of competitive learning that aims to construct basis vectors that efficiently span the sensorimotor space encountered by the robot. After training, the robot navigates from arbitrary locations to a desired goal location based on motor output vectors computed by indexing into sensorimotor memory with present as well as past perceptions, and using a saliency-based weighted averaging scheme for interpolating output vectors. By using past sensory inputs to provide the necessary context for disambiguating potentially similar perceptions, we alleviate the well-known problem of perceptual aliasing [5, 38] in non-Markov environments. We provide simulation results for a mobile robot, equipped with simple photoreceptors and infrared receivers, navigating within an enclosed obstacle-ridden arena. The method is shown to perform successfully in a variety of navigational tasks, some of which exhibit substantial perceptual aliasing.

2 Sparse Distributed Memory

One possible method for memory-based navigation is to use an associative memory in the form of a look-up table that associates a large set of perception vectors from different locations with corresponding set of actions required to navigate to a desired location [25]. However, such an approach suffers from at least three drawbacks: (a) the address space formed by the perception vectors is usually quite large and therefore, storing fixed reference vectors for every possible scenario becomes infeasible; (b) the training time increases drastically as the look-up table size increases, and (c) simple look-up table strategies relying on nearest-neighbor methods usually fail to generate the appropriate responses to novel situations. In this section, we address problems (a) and (b) by using only a *sparse* subset of the perceptual address space. This naturally leads to a memory known as sparse distributed memory (SDM) that was originally proposed by Kanerva [14]. We address the second problem of generalization in novel scenarios in the next section where we propose a modified form of SDM that uses competitive learning to adapt its sensorimotor space and radial interpolation to compute motor output vectors.

2.1 *Kanerva's Model*

Sparse distributed memory (SDM) (Figure 1) was first proposed by Kanerva [14] as a model of human long-term memory. It is based on the crucial observation that if concepts or objects of interest are represented by high-dimensional vectors, they can benefit from the very favorable matching properties caused by the inherent tendency toward orthogonality in high-dimensional spaces.

Kanerva's SDM can be regarded as a generalized random-access memory wherein the memory addresses and data words come from high-dimensional vector spaces. As in a conventional random-access memory, there exists an array of storage locations, each identified by a number (the address of the location) with associated data being stored in these locations as binary words. However, unlike conventional random-access memories which are usually concerned with addresses only about 20 bits long (memory size = 2^{20} locations) and data words only about 32 bits long, SDM is designed to work with address and data vectors with much larger dimensions. Due to the astronomical size of the vector space spanned by the address vectors, it is physically impossible to build a memory containing every possible location of this space. However, it is also unnecessary since only a subset of the locations will ever be used in any application. This provides the primary motivation for Kanerva's model: *only a sparse subset of the address space is used for identifying data locations and input addresses are not required to match stored addresses exactly but to only lie within a specified distance of an address to activate that address.*

The basic operation of SDM as proposed by Kanerva can be summarized as follows:

1. **Initialization**: The physical locations in SDM correspond to the rows of an $m \times k$ contents matrix $\mathbf{C}$ (initially filled with zeroes) in which data vectors $\in \{-1, 1\}^k$ are to be stored (see Figure 1 (a)). Pick m unique addresses (p-element binary vectors $\mathbf{r}_i$) at random for each of these locations (these addresses are represented by the matrix $\mathbf{A}$ in Figure 1 (a)).

2. **Data Storage**: Given an p-element binary address vector $\mathbf{r}$ and a k-element data vector $\mathbf{d}$ for storage, select all storage locations whose addresses lie within a Hamming distance of D from $\mathbf{a}$ (these activated locations are given by the *select* vector $\mathbf{s}$ in Figure 1 (a)). *Add* the data vector $\mathbf{d}$ to the previous contents of each of the selected row vectors of $\mathbf{C}$. Note that this is different from a conventional memory where addresses need to exactly match and previous contents are overwritten with new data.

3. **Data Retrieval**: Given an p-element binary address vector $\mathbf{r}$, select all storage locations whose addresses lie within a Hamming distance of D from $\mathbf{r}$ (these locations are again given by the vector $\mathbf{s}$). Add the values of these selected locations in parallel (*i.e.* vector addition) to yield a sum vector $\mathbf{S}$ containing the k sums. Threshold these k sums at 0 to obtain the data vector $\mathbf{d}'$ *i.e.* $d_i = 1$ if $s_i > 0$ and $d_i = -1$ otherwise.

Note that the addition step in (2) above is essentially a *Hebbian learning rule*. The statistically reconstructed data vector $\mathbf{d}'$ should be the same as the original data vector provided the *capacity* of the SDM [15] has not been exceeded. The intuitive reason for this is as follows: When storing a data vector $\mathbf{d}$ using an p-dimensional address vector $\mathbf{r}$, each of the selected locations receives one copy of the data. During retrieval with an address close to $\mathbf{r}$, say $\mathbf{r}'$, *most* of the locations that were selected with $\mathbf{r}$ are also selected with $\mathbf{r}'$. Thus, the sum vector contains most of the copies of $\mathbf{d}$, plus copies of other different words; however, due to the orthogonality of the address space

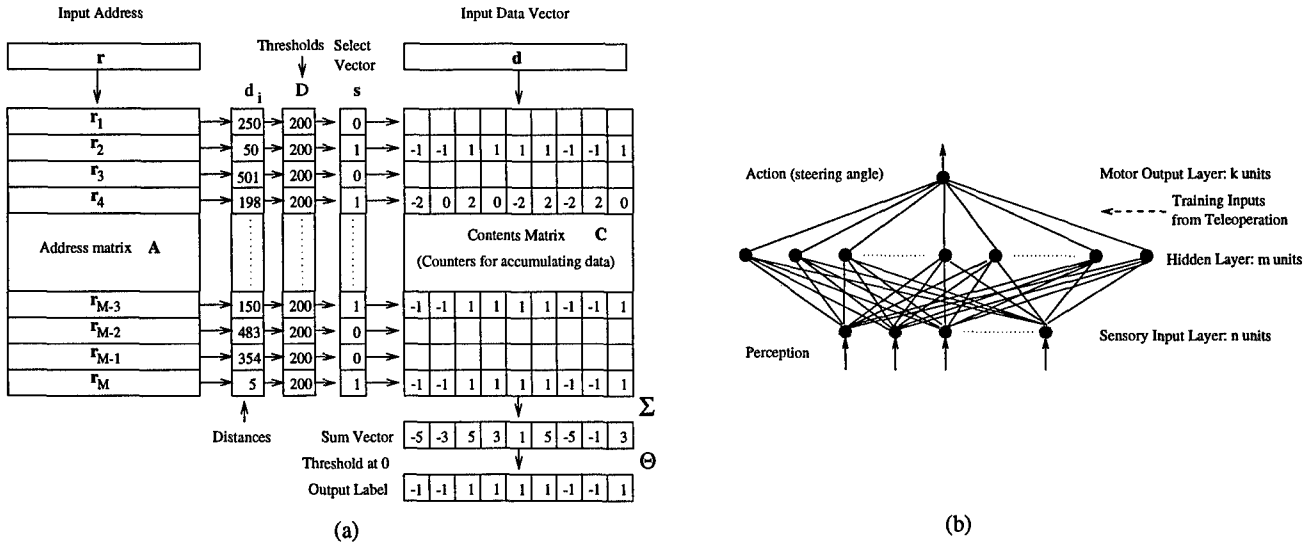


Figure 1: **Kanerva's Sparse Distributed Memory (SDM).** (a) shows a schematic depiction of the model as proposed by Kanerva for storage of binary (and bipolar) vectors. (b) shows a realization of the memory in the form of a three-layered fully connected network. The labels describe how the network can be used for perception-based navigation after suitable modifications (see Section 3).

for large p, these extraneous copies are much fewer than the number of copies of $\mathbf{d}$. This biases the sum vector in the direction of $\mathbf{d}$ and hence, $\mathbf{d}$ is output with high probability. A more rigorous argument based on signal-to-noise ratio analysis can be found in [15].

2.2 *Limitations of Kanerva's Model*

The model of sparse distributed memory as originally proposed by Kanerva has several weaknesses that prevent its direct use in memory-based navigation:

- Both the address and data vectors are required to be binary in the standard model of the SDM. Since most natural environments yield multivalued input patterns, we must either modify the indexing mechanisms of the model or recode all inputs into binary form. We chose the former option since the latter sacrifices the interpolation properties of the memory.
- The standard model of the SDM assumes a *uniform distribution* for the input address vectors whereas in most natural circumstances, the input address vectors tend to be clustered in many correlated groups distributed over a large portion of the multidimensional address space. Therefore, if addresses are picked randomly as suggested by Kanerva, a large number of locations will never be activated while a number of locations will be selected so often that their contents will resemble noise. We remedy this situation by allowing the input address space to *self-organize* according to the sensorimotor input distribution.
- The standard SDM uses a single fixed threshold D for activating address locations. While this simplifies the analysis of the memory considerably, it also results in poor performance since the actual values of the distances between an input address vector and the basis address vectors in the SDM are lost during quantization to 0 or 1. Our model uses *radial interpolation* functions that weight corresponding data vectors according to the address vector's closeness to the input vector.

3 A Self-Organizing SDM for Perception-Based Navigation

In the following, we describe the operation of a modified form of SDM that is suitable for memory-based navigation. The memory can be realized as a three-layer fully-connected feedforward network as shown in Figure 1 (b). Assume the memory contains m storage locations. The first layer consists of n units representing the input perception vector $\mathbf{p}$. The hidden layer consists of m units while the output layer is represented by k units. Let $\mathbf{w}_i$ $(1 \leq i \leq m)$ represent the vector of weights between hidden unit i and the input layer, and let $\mathbf{m}_i$ represent the vector of weights between hidden unit i and the output layer. The memory accepts multivalued perception vectors $\mathbf{p}$ from an arbitrary distribution and stores an associated multivalued motor vector $\mathbf{m}$ in a distributed manner in the data space.

3.1 *Initialization*

Pick m unique addresses (n-dimensional vectors $\mathbf{p}_i$) at random for each of the locations. This corresponds to randomly initializing the input-hidden weight vectors $\mathbf{w}_i$ $(1 \leq i \leq m)$.

3.2 Competitive Learning of Sensorimotor Basis Functions

Given an input perception vector $\mathbf{p}$ and an associated motor vector $\mathbf{m}$ during the training phase, we self-organize the address/data space using a *soft competitive learning rule* [26, 40]:

1. Calculate the Euclidean distances $d_j = \|\mathbf{w}_j^t - \mathbf{p}\|$ between $\mathbf{w}_j^t$ and the input perception vector $\mathbf{p}$.
2. Adapt all weight vectors (sensory address space vectors) according to:

$$\mathbf{w}_j^{t+1} \leftarrow \mathbf{w}_j^t + g_j(t)P_t(d_j)(\mathbf{p} - \mathbf{w}_j^t) \quad (1)$$

where P_t is defined as:

$$P_t(d_j) = \frac{e^{-d_j^2/\lambda_j(t)}}{\sum_{k=1}^{m} e^{-d_k^2/\lambda_k(t)}} \quad (2)$$

and g_j is given by $g_j(t) = 1/n_j(t)$ where the counter:

$$n_j(t+1) = n_j(t) + P_t(d_j) \quad (3)$$

$P_t(d_j)$ can be interpreted as the probability of the prototype vector $\mathbf{w}_j$ winning the current competition for perception $\mathbf{p}$. Note that the probability vector $\mathbf{P}$ obtained by vectorizing the $P_t(d_j)$ for $1 \leq j \leq m$ is the equivalent of the *select* vector $\mathbf{s}$ in Kanerva's model (Figure 1 (a)).

The "temperature" parameter $\lambda_j(t)$ is gradually decreased to a small value in a manner reminiscent of simulated annealing. This causes the learning algorithm to evolve from an initially soft form of competition with a large number of winners to the case where only a *sparse* number of winners exist for any given input vector. The soft competition in the beginning tunes the initially random prototype vectors towards the input sensory space (thereby preventing the occurrence of "dead" units which never get updated) while the later existence of sparse number of winners helps in fine tuning the prototype vectors to form a set of distributed *basis vectors* spanning the sensory input space.

3. Given a training motor input $\mathbf{m}$, adapt the prototype vectors stored in the motor space according to equation 1 using the *same* values for d_j as in 1 above (*i.e.* distance between the input perception vector and the sensory basis vectors $\mathbf{w}_j^t$):

$$\mathbf{m}_j^{t+1} \leftarrow \mathbf{m}_j^t + \beta_j(t)P_t(d_j)(\mathbf{m} - \mathbf{m}_j^t) \quad (4)$$

where $\beta_j(t)$ $(0 < \beta_j(t) < 1)$ is a gain function. Note that $\beta_j(t)$ *does not necessarily* have to decrease with time (this reinforcement strength could be made to depend on other factors such as importance to the animate system as evaluated by other modalities or other internal value mechanisms).

3.3 Computing Motor Output Vectors

Following training, the memory *interpolates* between the stored motor basis vectors to produce a motor output vector $\mathbf{o}$ for a given perception vector $\mathbf{p}$, as follows:

1. Calculate the Euclidean distances d_j between $\mathbf{w}_j$ and the input perception vector $\mathbf{p}$.
2. Let m_{ji} $(1 \leq i \leq k)$ denote the weight from hidden unit j to output unit i. Then, the ith component of the reconstructed output vector $\mathbf{o}$ (in other words, the output of the output unit i) is given by:

$$o_i = \sum_{j=1}^{m} P_t(d_j)m_{ji} \quad (5)$$

The saliency-based weighted averaging above is a form of normalized radial interpolation that is similar to the output operation of *radial basis function* (RBF) networks [28]: the closer the current perception is to a given sensory basis vector, the more "salient" that memory location and the more the weight assigned to the motor vector associated with that basis vector. The above scheme is inspired by recent neurophysiological evidence [23] that the *superior colliculus*, a multilayered neuronal structure in the brain stem that is known to play a crucial role in the generation of saccadic eye movements, in fact employs a population averaging scheme similar to the one above to compute saccadic motor vectors.[1]

4 Predictive Sensorimotor Memory

The self-organizing SDM described in the preceding section has been shown to be useful for perceptual homing by an autonomous mobile robot [35]. However, it is not hard to see that the above method is limited to only those navigational behaviors that can be modeled by *Markov* processes: the current motor output is dependent solely on the current perception and past inputs are treated as irrelevant for determining current action. In general environments, however, a Markovian assumption is often inappropriate, and any method that relies on such an assumption suffers from the problem of *perceptual aliasing* [5, 38].

4.1 Perceptual Aliasing

Perceptual aliasing, a term coined in [38], refers to the situation wherein two or more identical perceptual inputs require different responses from an autonomous system. A number of factors such as limited sensing capability, noisy sensory inputs, and restricted resolution in addition to inherent local ambiguity in typical environments contribute towards exacerbating perceptual aliasing. The effects of aliasing can be reduced to some extent by incorporating additional sensory information that suffice to disambiguate between any two given

[1] We refer the interested reader to an earlier paper [32] for a closely related method for visuomotor learning of saccadic eye movements for a robot head.

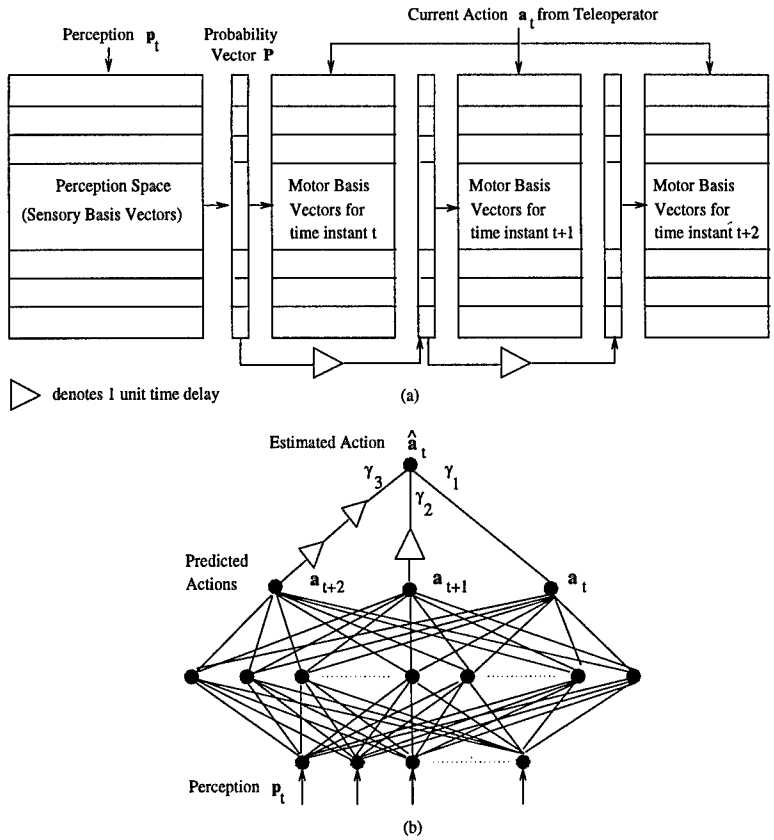


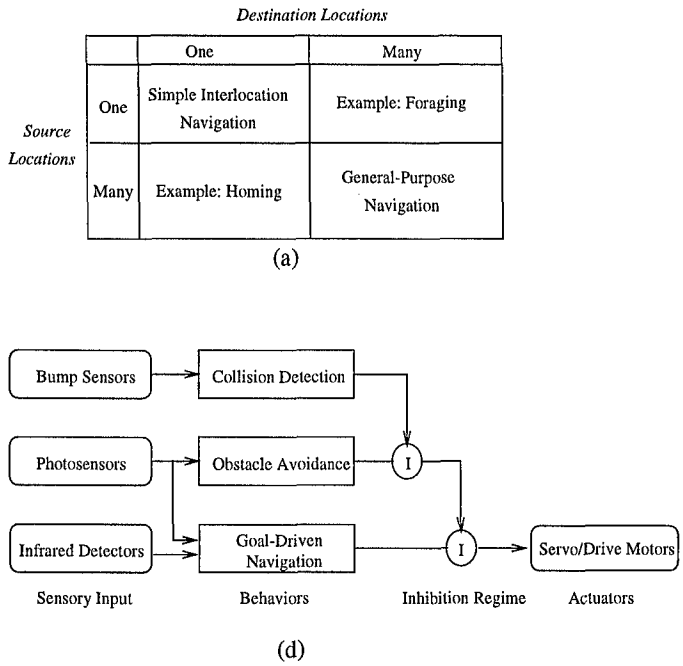
Figure 2: **Predictive Sparse Distributed Memory.** (a) shows the procedure of training the predictive memory. The current motor input is fed to each of the motor memories which are indexed by the current perception (left), the previous one (middle), and the perception before the previous one (right). This allows the memory to generate predictions of motor output for the next two time steps in addition to the current one during the navigation phase. (b) shows the memory (in its network form) during the navigation phase. The current estimate of motor output $\hat{\mathbf{a}}_t$ is computed by averaging over the motor action vectors for the current time instant t predicted by current and past perceptual inputs.

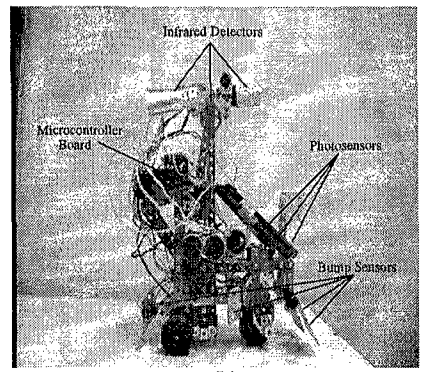
situations [38, 39]. However, these methods still rely only on current percepts and thus, are unable to overcome aliasing in non-Markov environments. An alternative approach that is tailored toward modeling non-Markov sensorimotor processes is to base the current output on both the current as well as the sequence of past outputs (within a certain time window). In other words, the current percept not only predicts the action for the current time instant t but also actions for future time instant $t, t+1, \ldots, t+k$, where k is a parameter whose value can be either predetermined or adapted on-line to counteract the effects of aliasing. The motor action for the current time instant t is then determined by a weighted average of the actions recommended by the current as well as past perceptions upto time $t-k$. This solution shares some similarities with Kanerva's k-fold memory for storage and retrieval of sequences [14]; however, all the weaknesses inherent in Kanerva's original model (Section 2.2) still apply to the k-fold memories, thus preventing their direct application in countering the aliasing problem.

4.2 *Using Past Perceptions for Motor Prediction*

The informal solution for perceptual aliasing sketched in the previous section can be formalized as follows. Let $\mathbf{p}_t$ be the current perception vector and let $\mathbf{m}$ be the current motor action vector. Figure 2 shows the predictive memory. Note that the motor space now contains additional sets of connections between the hidden and output layer, each such set determining the action to be executed at a given time in the future. A further set of connections (see (b)), some of which involve time delays of different durations, appropriately combine the outputs of this intermediate layer to yield an estimate of current motor output. The operation of the predictive memory is as follows:

- **Training**: During training (Figure 2 (a)), the perception space is self-organized as given by Equation 1. The motor space is also self-organized as in Equation 4 but using the *current* motor vector as the training signal for *each* set of hidden to motor output unit connections, with the constraint that the probability weight vector $\mathbf{P}$ (see Section 3.2) for the set of connections determining the action for time $t+i$ is the one that was obtained by indexing into the sensory address space using the perception at time $t-i$. Such a training paradigm ensures that after training, the perception at time t generates predicted actions for time steps $t, t+1, \ldots, t+k$.





(b)

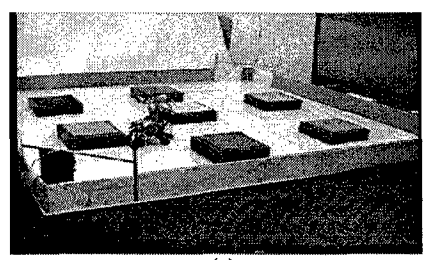

(c)

Figure 3: **Experimental Methodology**. (a) A hierarchical decomposition of the general navigation problem. Preliminary experimental results indicate that the robot is capable of acquiring each of the four cases of navigational behaviors shown in the diagram. (b) The mobile robot that inspired the simulations in the experiments. (c) The robot in its environment. (c) Robot control architecture: the collision detection and obstacle avoidance routines were autonomously learned on-line by the robot as described in [10]. "I" denotes inhibition of the current behavior by a lower-level behavior.

- **Navigation**: During the autonomous navigation phase (Figure 2 (b)), the current perception yields a set of actions $\mathbf{a}_t, \mathbf{a}_{t+1}, \ldots, \mathbf{a}_{t+k}$. Thus, at any given time instant t ($t \geq k$), there exist k estimates $\mathbf{a}_t^1, \mathbf{a}_t^2, \ldots, \mathbf{a}_t^k$ of what the current action should be based on current and past sensory inputs. We obtain the combined estimate of current motor output using the following weighted averaging method:

$$\hat{\mathbf{a}}_t = \sum_{i=1}^{k} \gamma_i \mathbf{a}_t^i \qquad (6)$$

where the γ_i determine the fidelity of estimate i and can be experimentally determined or autonomously learned.

5 Experimental Results

The methods proposed in the previous sections were tested by simulating the behavior of an actual robot (Figure 3 (b)) in an enclosed obstacle-filled environment (Figure 3 (c)). This robot was previously used for on-line learning of a hierarchical set of behaviors [10] but the slow processing speed of the on-board microcontroller unfortunately limited its use in the present endeavor; we therefore evaluated the feasibility of the algorithms presented in this paper by using a simulation of the robot instead. The simulated robot was equipped with the same three classes of sensors as did its real world counterpart:

- **Bump Sensors**: Realized using digital microswitches, these sensors indicate whether the robot was physically touching an obstacle. Five of these sensors, placed at different locations around the robot, were used for the *obstacle detection* behavior.

- **Photosensors**: Six horizontally-positioned photoreceptors (implemented via shielded photoresistors) were employed for measuring the amount of light from a light source located near the arena. Six tilted photoreceptors were used for measuring light intensity value due to the color of the floor and for detecting surrounding obstacles. The outputs from these sensors were used for the *obstacle avoidance* behavior as well as for learning the perception-based navigational behaviors;

- **Infrared detectors**: These sensors, when used in conjunction with infrared detection software, indicate the strength of the modulated infrared light in a small spread along the line of sight of the sensor. Output from four of these sensors were used for learning the navigational behaviors.

The above sensory repertoire was supplemented by two effectors consisting of a rear drive motor and a servo motor at the front for steering the robot. For the simulations, robot perceptions were computed using the simplifying assumption that light/infrared intensity at a particular orientation θ and at a distance r from the source is proportional to the solid angle subtended by the source (*i.e.* $cos(\theta)/r^2$), assuming unit area for the robot receptors.

In the first set of experiments, the simple non-predictive self-organizing SDM from Section 3 was used for training the robot to home to a particular location from an arbitrary

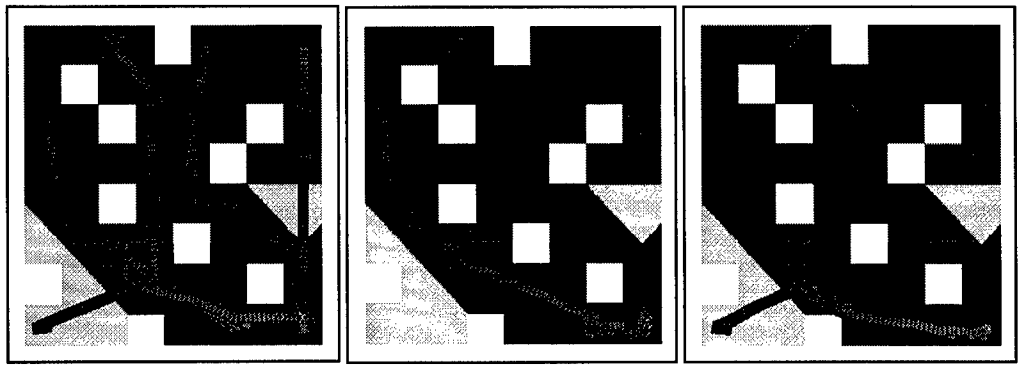


Figure 4: **Homing using the non-predictive Sparse Distributed Memory** (a) The paths on which the network was trained by teleoperating the robot. Floor color is depicted by shades of grey and obstacles are colored white. The home position is marked by an 'X'. (b) and (c) depict the paths chosen by the robot when placed at two different positions within the arena. Note the slight deviations from the training paths caused by mild perceptual aliasing; the obstacle avoidance behavior is automatically invoked when the deviations cause encounters with the wall or the square obstacles.

number of other locations in the robot arena (this corresponds to the many-one navigation box in the classification of Figure 3 (a)). Figure 4 shows some typical examples where the robot successfully navigates to the home position. Note that the existence of mild perceptual aliasing causes it to deviate on some occasions, thereby necessitating the use of the lower level obstacle avoidance behaviors.

In the second set of experiments (corresponding to the many-many box in Figure 3 (a)), the robot was trained on a number of paths that *intersected* each other at different locations (Figure 5 (a)). Thus, at these locations, local perceptions alone do not suffice to determine the steering direction, and the non-predictive memory usually fails to find the correct direction to continue in order to reach the pertinent goal destination. On the other hand, as shown in Figure 5 (b), (c) and (d), the predictive SDM allows the robot to use the context of past perceptions to determine its current action, thereby guiding it to its appropriate destination in spite of the presence of varying degrees of perceptual aliasing.

6 Summary and Conclusions

We have shown that a predictive sparse distributed memory provides an efficient platform for learning adaptive navigational behaviors. The proposed method enjoys severable favorable properties:

- **Sparse Memory**: In contrast to nearest-neighbor look-up table techniques, the present method employs only a sparse number of memory locations and avoids the "curse of dimensionality" problem by intelligently sampling the high-dimensional sensorimotor space using competitive learning (see below).
- **Competitive Learning of Sensorimotor Basis Functions**: The contents of the sparse memory are self-organized using a form of competitive learning that can be related to maximum likelihood estimation [26]. The learning rule allows the memory to autonomously form its own set of basis functions for describing the current sensorimotor space.
- **Distributed Storage**: Inputs to the memory are distributed across a number of locations, thereby inheriting the well-known advantages of a distributed representation [11] such as generalization to previously unknown inputs and resistance to faults in memory and internal noise.
- **Motor Prediction based on Past Perceptual History**: The problem of *perceptual aliasing* is alleviated by employing past perceptions to predict and influence current motor output. This extends the application of the method to non-Markov sensorimotor environments and distinguishes our approach from previous neural network approaches based on training procedures such as back-propagation [29].
- **Biological Plausibility**: The structure of the memory bears some striking similarities to the organization of the mammalian cerebellum (and therefore to the cerebellar model of Marr [21] and the CMAC of Albus [1]).[2] It is therefore plausible that biological structures and learning processes similar in spirit to those proposed herein may underlie goal-directed perception-based navigation in animals.

Sparse distributed memories have previously been used for a wide variety of tasks such as object recognition [34], face recognition [33], speech recognition [31], speech synthesis [12], and weather prediction [36]. The present work shows that with suitable modifications, such memories can be used for learning useful navigational behaviors in mobile robots as well. Ongoing work involves implementing the learning method on a recently acquired wheelchair robot and designing learning procedures for on-line adaptation of some of the free parameters in the current method such as the length of the perceptual history window and the fidelity γ_i for combining

[2] See [15, 35] for more details.

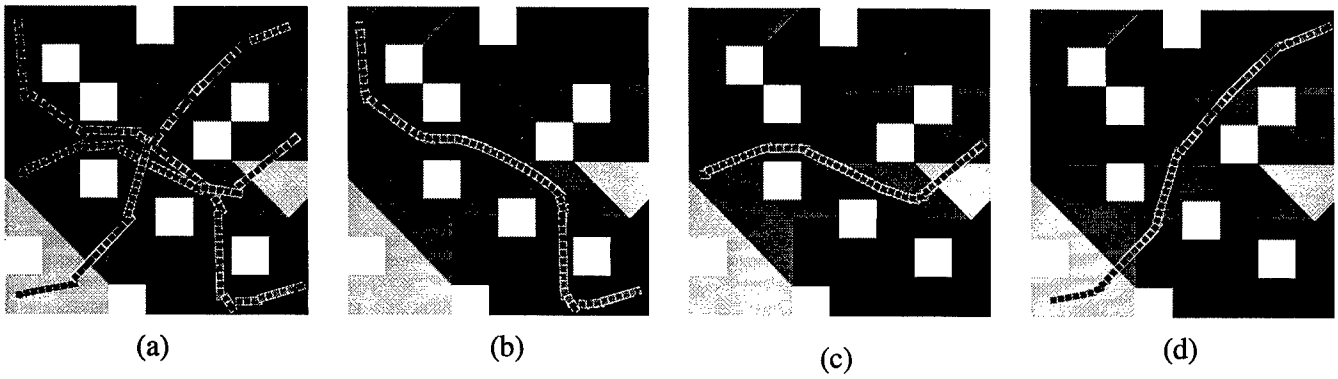


Figure 5: **Navigation using the Predictive Sparse Distributed Memory** (a) The paths on which the network was trained by teleoperating the robot. Note the intersection of the training paths at various points which gives rise to perceptual aliasing. (b), (c) and (d) show that the predictive memory is able to circumvent aliasing effects and follow the path to its goal destination by using past sensory information as a contextual aid to disambiguate aliased perceptions.

predicted actions. A simultaneous effort involves exploring the use of predictive recurrent networks in conjunction with learning rules other than competitive learning for generating useful behaviors in autonomous robots.

Acknowledgments

We would like to thank Dana Ballard for several interesting discussions, Chris Brown for his encouragement and support for building the mobile robot, and Mike Van Wie for his help in the construction of the robot.

References

[1] James S. Albus. A theory of cerebellar functions. *Math. Biosci.*, 10:25–61, 1971.

[2] M. Asada, E. Uchibe, S. Noda, S. Tawaratsumida, and K. Hosoda. Coordination of multiple behaviors acquired by a vision-based reinforcement learning. In *Proceedings of the 1994 IEEE/RSJ International Conference on Intelligent Robots an Systems*, pages 917–924, Munich, Germany, 1994.

[3] R.D. Beer and J.C. Gallagher. Evolving dynamical neural networks for adaptive behavior. *Adaptive Behavior*, 1:91–122, 1992.

[4] Rodney A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2(1):14–22, April 1986.

[5] Lonnie Chrisman. Reinforcement learning with perceptual aliasing. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, 1992.

[6] Dave Cliff, Philip Husbands, and Inman Harvey. Evolving visually guided robots. In J. A. Meyer, H. Roitblat, and S. Wilson, editors, *From Animals to Animats 2: Proceedings of the Second International Conference on the Simulation of Adaptive Behavior*, pages 374–383. Cambridge, MA: MIT Press, December 1992.

[7] Jonathan H. Connell. *Minimalist mobile robotics : A colony-style architecture for an artificial creature*. Academic Press, Boston, MA, 1990.

[8] P.G.R. de Bourcier. Animate navigation using visual landmarks. Cognitive Science Research Papers 277, University of Sussex at Brighton, May 1993.

[9] A. Elfes. Sonar-based real-world mapping and navigation. *IEEE Journal of Robotics and Automation*, 3:249–265, 1987.

[10] Olac Fuentes, Rajesh P. N. Rao, and Michael Van Wie. Hierarchical learning of reactive behaviors in an autonomous mobile robot. In *Proceedings of the IEEE International Conf. on Systems, Man, and Cybernetics*, 1995.

[11] G.E. Hinton, J.L. McClelland, and D.E. Rumelhart. Distributed representations. In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 1. MIT Press, Cambridge, MA, 1986.

[12] U.D. Joglekar. Learning to read aloud: A neural network approach using sparse distributed memory. Technical Report 89.27, Research Institute for Advanced Computer Science, NASA Ames Research Center, 1989.

[13] Leslie P. Kaelbling. *Learning in embedded systems*. MIT Press, Cambridge, MA, 1993.

[14] P. Kanerva. *Sparse Distributed Memory*. Cambridge, MA: Bradford Books, 1988.

[15] Pentti Kanerva. Sparse distributed memory and related models. In Mohamad H. Hassoun, editor, *Associative Neural Memories*, pages 50–76. New York: Oxford University Press, 1993.

[16] Ben J. A. Krose and Marc Eecen. A self-organizing representation of sensor space for mobile robot navigation. In *IEEE/RSJ/GI International Conference on Intelligent Robots an Systems*, pages 9–14, 1994.

[17] Benjamin J. Kuipers and Yung-Tai Byun. A robust, qualitiative approach to a spatial learning mobile robot. In *SPIE Cambridge Symposium on Optical and Optoelectronic Engineering: Advances in Intelligent Robotics Systems*, November 1988.

[18] M. A. Lewis, A. H. Fagg, and A. Solidum. Genetic programming approach to the construction of a neural network for control of a walking robot. In *Proceedings of the 1992 IEEE International Conference on Robotics and Automation*, Nice, France, 1992.

[19] Pattie Maes and Rodney A. Brooks. Learning to coordinate behaviors. In *Proceedings of AAAI-90*, pages 796–802, 1990.

[20] Sridhar Mahadevan and Jonathan Connell. Scaling reinforcement learning to robotics by exploiting the subsumption architecture. In *Proceedings of the Eighth International Workshop on Machine Learning*, 1991.

[21] David Marr. A theory of cerebellar cortex. *J. Physiol. (London)*, 202:437–470, 1969.

[22] Maja Mataric. Integration of representation into goal-driven behavior-based robot. *IEEE Transactions on Robotics and Automation*, 8(3):304–312, 1992.

[23] James T. McIlwain. Distributed spatial coding in the superior colliculus: A review. *Visual Neuroscience*, 6:3–13, 1991.

[24] Ulrich Nehmzow and Tim Smithers. Mapbuilding using self-organizing networks in "Really Useful Robots". In J.-A. Meyer and S. W. Wilson, editors, *From Animals to Animats 1: Proceedings of the First International Conference on Simulation of Adaptive Behavior*, pages 152–159. Cambridge, MA: MIT Press, 1991.

[25] Randal. C. Nelson. Visual homing using an associative memory. *Biological Cybernetics*, 65:281–291, 1991.

[26] Steven J. Nowlan. Maximum likelihood competitive learning. In *Advances in Neural Information Processing Systems 2*, pages 574–582. Morgan Kaufmann, 1990.

[27] David Pierce and Benjamin Kuipers. Learning hill-climbing functions as a strategy for generating behaviors in a mobile robot. In *From Animals to Animats: Proceedings of the First International Conference on Simulation of Adaptive Behavior*, pages 327–336, 1991.

[28] T. Poggio and F. Girosi. Networks for approximation and learning. *Proc. IEEE*, 78:1481–1497, 1990.

[29] D.A. Pomerleau. ALVINN: An autonomous land vehicle in a neural network. In D.S. Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 1, pages 305–313. Morgan Kaufmann, San Mateo, 1989.

[30] D.A. Pomerleau. Efficient training of artificial neural networks for autonomous navigation. *Neural Computation*, 3(1):88–97, Spring 1991.

[31] R.W. Prager and F. Fallside. The modified Kanerva model for automatic speech recognition. *Computer Speech and Language*, 3(1):61–81, 1989.

[32] Rajesh P.N. Rao and Dana H. Ballard. Learning saccadic eye movements using multiscale spatial filters. In G. Tesauro, D.S. Touretzky, and T.K. Leen, editors, *Advances in Neural Information Processing Systems 7*, pages 893–900. Cambridge, MA: MIT Press, 1995.

[33] Rajesh P.N. Rao and Dana H. Ballard. Natural basis functions and topographic memory for face recognition. In *Proc. of IJCAI*, pages 10–17, 1995.

[34] Rajesh P.N. Rao and Dana H. Ballard. Object indexing using an iconic sparse distributed memory. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 24–31, 1995.

[35] Rajesh P.N. Rao and Olac Fuentes. Perceptual homing by an autonomous mobile robot using sparse self-organizing sensory-motor maps. In *Proceedings of World Congress on Neural Networks (WCNN)*, pages II380–II383, 1995.

[36] David Rogers. Predicting weather using a genetic memory: A combination of Kanerva's sparse distributed memory and Holland's genetic algorithms. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems 2*, pages 455–464. Morgan Kaufmann, 1990.

[37] Jun Tani and Naohiro Fukumura. Learning goal-directed sensory-based navigation of a mobile robot. *Neural Networks*, 7(3):553–563, 1994.

[38] Steven D. Whitehead and Dana H. Ballard. Learning to perceive and act by trial and error. *Machine Learning*, 7(1):45–83, 1991. (Also Tech. Report # 331, Department of Computer Science, University of Rochester, 1990.).

[39] Lambert Wixson. Scaling reinforcement learning techniques via modularity. In *Proceedings of the Eighth International Workshop on Machine Learning*, pages 368–372. Morgan Kaufmann, 1991.

[40] E. Yair, K. Zeger, and A. Gersho. Competitive learning and soft competition for vector quantizer design. *IEEE Trans. Signal Processing*, 40(2):294–309, 1992.

EVOLUTION

A Developmental Model for the Evolution of Complete Autonomous Agents

Frank Dellaert [1] and Randall D. Beer [2]

[1] Dept. of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213
[2] Dept. of Computer Engineering and Science, Dept. of Biology
Case Western Reserve University, Cleveland, OH 44106
[2] Santa Fe Institute, 1399 Hyde Park Rd, Santa Fe, NM 87501
e-mail: dellaert@cs.cmu.edu, beer@alpha.ces.cwru.edu

Abstract

Development is an important, powerful and integral element of biological evolution. In this paper we present two models of development that can be used to evolve functional autonomous agents, complete with bodies and neural control systems. The first and most complex model is more biologically defensible in its details. It has been used to hand-design a genome for the development of complete agents capable of executing a simple avoidance task. These agents were then incrementally improved through evolution. The second model is simpler and uses a random Boolean network model for the genome and cell state that is somewhat more removed from the biological realm, making it easier to analyze and more amenable to artificial evolution. Using this model, we have successfully evolved complete agents from scratch that are capable of following curved lines.

1. Introduction

In this paper we present a model of development that has been used to evolve functional autonomous agents, complete with a morphological structure and a neural control system. Earlier work that involved a more biologically defensible but more complex model is contrasted with a new and simplified approach that performs surprisingly better.

Development is an important and integral part of biological evolution. Genetic changes are not directly manifested in phenotypic changes, as is often assumed both in population genetics and in most autonomous agent work involving evolution. Rather, a complex developmental machinery mediates between genetic information and phenotype, and this has many consequences. It provides a certain robustness by filtering out genetic changes (i.e. some genetic changes make little or no difference to the final phenotype; there is an equifinality to development). It also provides a natural way to try out a spectrum of mutations, i.e. the same type of genetic mutation can produce anything from no effect to a very large effect in the phenotype, depending on when the affected gene acts during development. Furthermore, development provides a compact genetic encoding of complex phenotypes, allows incremental building of complex organisms, and supports symmetry and modular designs.

For these reasons, there is a growing interest in modeling development (Lyndenmayer and Prusinkiewicz 1989; Wilson 1989; Mjolsness, Sharp and Reinitz 1991; deBoer, Fracchia and Prusinkiewicz 1992; Fleischer and Barr 1994; Kitano 1994). Many ongoing efforts aimed at including simple developmental models in evolutionary simulations can be found in the literature (Belew 1993; Cangelosi, Parisi and Nolfi 1993; Gruau and Whitley 1993; DeGaris 1994; Kodjabachian and Meyer 1994; Nolfi, Miglino and Parisi 1994; Sims 1994; Jakobi 1995).

Much of the latter work has focused on modeling *neural* development, however. But biological bodies and nervous systems co-evolve. Body morphology and nervous systems can constrain and shape one another. Somatic and genetic factors can interact, and this occurs not only during development, but on an evolutionary scale as well. Problems posed by evolution can be solved by a combination of body and neural changes. Allowing both body and nervous system to co-evolve can provide a smoother and more incremental path for substantial changes.

Also, most of this work is highly abstracted from biological development (e.g. using grammars). While there are good reasons for this (familiarity, simplicity, computational speed, emphasis on performance not biology, etc.), too little is currently understood about development to know what are the right abstractions to make. Development completely transforms the structure of the space that is being searched. If we're lucky, this transformation will allow us to evolve interesting agents more easily. But if we're unlucky, we could actually make the search problem *harder*. Because so little is currently understood about the overall 'logic' of development, it is important that we explore many different levels of abstraction to get a sense of the tradeoffs involved. An important aspect of this exploration should be to explore developmental models that are more biologically realistic in their basic structure than the highly abstract models that have currently been explored. That is the aim of this paper.

In the next section we give a brief overview of the general approach that we have adapted to model a developmental process for autonomous agents. It is common to both models we discuss in the paper. Section 3 presents a biologically defensible model of development, complete with a mechanism for the emergence of a nervous system inspired by axonal growth cones. The expressiveness of the model is demonstrated by means of a hand-designed genome, able to direct the development of a functional and complete agent that can execute a simple task in a simulated world. In Section 4, we discuss a simplified model that addresses some of the problems of the earlier model, and show that it can be used to evolve functional agents from scratch. We

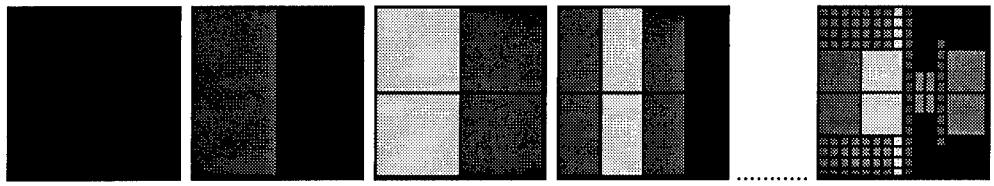


Figure 1: Starting with an 'egg cell' the developmental simulation yields a multicellular square as the adult organism.

show the example of an agent evolved to execute a line following task. Section 5 discusses some of the lessons we learned from this work, and suggests some avenues for further research.

2. Overview of the Developmental Model

In nature it is the performance of an adult organism in its environment that will determine whether its genetic material is propagated. However, in sharp contrast with the model usually assumed in the genetic algorithm literature, genes do not directly specify the traits of an animal. Rather, they specify the developmental sequence by which an animal grows out of a single egg cell to a fully developed phenotype. It is only after this process is complete that specific traits or behavior can be selected for or against[1].

For reasons specified in the introduction and elsewhere (Dellaert and Beer 1994a; Dellaert 1995), we believe that using a developmental model in conjunction with the genetic algorithm (GA) can help us to better evolve autonomous agents. Thus, we have tried to implement this in simulation, where a genetic algorithm will supply us with some genetic material, the genotype, that is then transformed by a developmental model into a fully grown organism, the phenotype. It is the performance of the phenotype that will determine whether its genotype is selected inside the GA.

In particular, we have implemented a model of development for simple simulated organisms that start out as a single cell but 'grow' into multicellular organisms. An example of one such developmental sequence is shown in figure 1. The fully developed agent will then be evaluated on how well it performs a simple task in a simulated world.

In contrast to the complex nature of real biological cells that are able to move and change their shape, our simulated cells are modeled as simple, rigid two-dimensional squares. This choice allowed us to implement cell-division in a particularly efficient way, keeping the computational cost of the simulation acceptable. Indeed, after two rounds of divisions, the resulting cells are again square cells. Our implementation does allow for more complex cell models to be substituted in place of the square cell model should that need arise in the future.

A range of different cell types coexist in any full-grown biological organism. This is so even though each cell possesses the same identical copy of the genome, a sequence of DNA unique to that particular animal. Yet, not all the cells have the same characteristics or behavior. Although the cells contain identical genes, different subsets of genes are expressed in different cells, giving them different properties.

In our model we have implemented a similar arrangement: each cell has an identical copy of a simulated genome inside it, but the subset of the 'genes' active at any given moment determines what type of cell it is and how the cell will behave during development. Exactly how this is implemented is at the core of both models we will present, and will be discussed in detail below.

During biological development, however, the subset of genes expressed in each cell is not static but rather in constant flux. The cell responds to its environment and to the instructions coded in its genome by changing its composition continuously, until it has differentiated fully into one of the 'adult' cell types. In turn, the instructions given by the genome are a function of the state of the cell. Thus, cell state and genome comprise an interwoven dynamical system, a genetic regulatory network, and it is the collective unfolding of the dynamics of many such systems -one for each cell- that constitutes development. On a higher level of abstraction, development can be seen as the sequence of events by which the cells in the body differentiate to perform the various functions inside the animal.

The heart of our developmental model is formed by exactly one such genome-state dynamical system that lives inside each of our model cells. The active subset of model genes inside a cell will be regarded as the cell state, and the possible state transitions from that state are governed by both the genome, the current state and the environment in which the cell finds itself. Qualitatively, this picture is similar to what we see in biological cells, albeit quite simplified in the details. Our developmental simulation can now similarly be viewed as the sequence of events by which the state of the cells differentiate from the initial 'egg cell' state -and from one another- to form a particular cell type arrangement that will suit a particular task.

As explained in the introduction, we are interested in having both a morphological component, i.e. the development of the physical extent of a simulated organism, and a neural component, i.e. how the nervous system of an organism develops. Together, these components should form a complete autonomous agent. To this end, we have associated certain simulated cell types with particular functions, e.g. sensors, interneurons, or actuators. We then also provide a model of how these control components get wired up in a working neural network i.e. the neural developmental component of the model.

This high level specification of the model needs to be complemented with quite a few implementation issues and choices to make it work. We need to specify which states will lead to cells dividing. Since we have chosen to update

[1]This is not entirely true: there will also exist mutations that prevent an organism from developing to maturity.

all the cell states synchronously, we need to break the symmetry after the first division to get interesting dynamical behavior. Some mechanism of intercellular communication must be implemented to make it possible for cells to influence each other's state. Meaning must be assigned to the different possible genes, so that we can interpret state as cell type. And finally, the detailed mechanism for neural development needs to fully specified.

In the following we will present two different implementations of the developmental model: a complex one and a simple one. The complex model is more biologically defensible, but suffers from its complexity. The simple model is further removed from biological reality in its implementation details, but has the advantage that it is computationally more tractable and easier to analyze. We will discuss both models in the next sections.

3. A Complex Model

The first implementation actually models simplified genome and cytoplasm entities inside each cell, and even has model 'proteins' that are produced by the genome and are collected inside the 'cytoplasm'. The set of proteins in each cell determines what kind of events the cell reacts to and which signals it emits while the developmental sequence unfolds. In addition, there is an elaborate model of neural development based on a growth-cone model that detects the presence of proteins in the cells and grows accordingly.

Genome-cytoplasm Model

In the complex model, each cell contains a 'cytoplasm' and a 'genome'. The cytoplasm contains 'proteins', and the proteins present in each cell determine its capabilities, i.e. cells with a different set of proteins can be thought of as having a different cell type. In this regard our model proteins can be thought of as having a similar role as biological proteins. We represent each model protein by a unique integer, and implemented the cytoplasm as a set of integers.

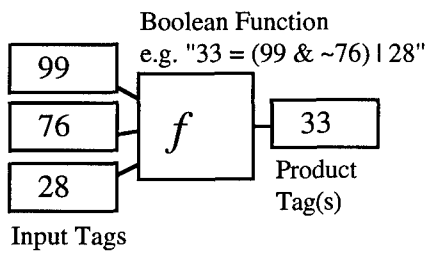


Figure 2: An artificial genome consists of artificial 'operons', one of which is shown here.

The genome consists of a set of 'operons', an example of which is shown in figure 2. As can be seen, each operon is made up of a set of input tags, a Boolean function and a set of output tags. In each time step of the simulation, the input tags determine the input to the Boolean function, and if the output of the Boolean function is evaluated to TRUE, the protein corresponding to the output tag is injected into the cytoplasm. An input tag will give a 1 to the Boolean function if its corresponding protein is present, and 0 otherwise.

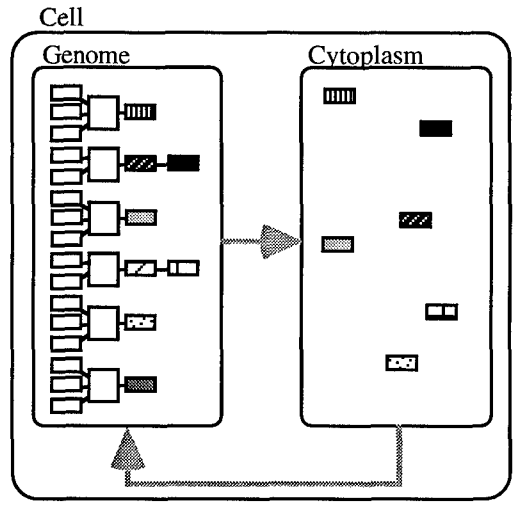


Figure 3: Schematic summary of the cytoplasm-genome model.

Thus, the content of the cytoplasm can be seen as the state of the cell or its cell type, and the genome will determine how this state will change over time. It is the interplay between genome and cytoplasm determines how the cell's cell type changes over time. This interplay is summarized graphically in figure 3.

Development

With the description of the cell's internals in the previous section, we can explain how the developmental simulation will proceed. The organism starts out as a single cell, with its genome given by the GA. The cytoplasm is initially the empty set. Then, for a constant number of iterations, all cells in the organism go through a simulated cell cycle in parallel. This 'cell cycle' consists of two phases. (1) During 'interphase' the cytoplasm is updated by evaluating all the operons in the genome, as explained above. (2) During mitosis, each cell checks for the presence of a special protein (which we will denote *tDividing*, where the leading t stands for 'protein tag') and divides if found.

The simulation ensures that the first cell always goes through division by injecting the *tDivision* protein into the cell's cytoplasm prior to starting the cell cycle. Thus, after the first division we end up with two cells, each with identical cytoplasm and genome that it inherited from their parent cell. This presents a problem: since all the cells obey the same deterministic rules set out above, the remainder of the simulation will yield identical cell types at each time step, and no interesting organisms will emerge.

Therefore we add two additional mechanisms, symmetry breaking and intercellular communication, to ensure that the developmental process exhibits interesting dynamics. Symmetry breaking happens just after the first division event and is implemented by injecting a special protein in only one of the first two daughter cells. This needs to happen only once: from now on these cells' descendants will also differ, as their dynamical trajectories start from different initial conditions. Cell communication, on the other hand, can happen at every cell cycle, and consists of a mechanism that allows one cell to cause the introduction of

a protein into another cell. This is modeled after the biological mechanism of induction, and involves proteins that represent morphogens, receptors and intracellular messenger proteins. The implementation details can be found in (Dellaert and Beer 1994a-b; Dellaert 1995).

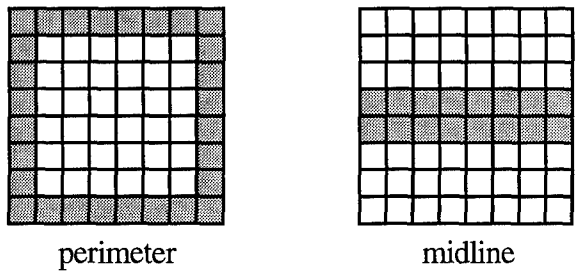


Figure 4: Two special receptors can lead to the differentiation of cells at the perimeter and adjacent to the midline of the organism, respectively.

The model for intercellular communication is also used to introduce two supplementary features that establish a perimeter and midline on the organism that can be detected by the cells, which can lead to local differentiation of cells as seen in figure 4.

A Detailed Neural Developmental Model

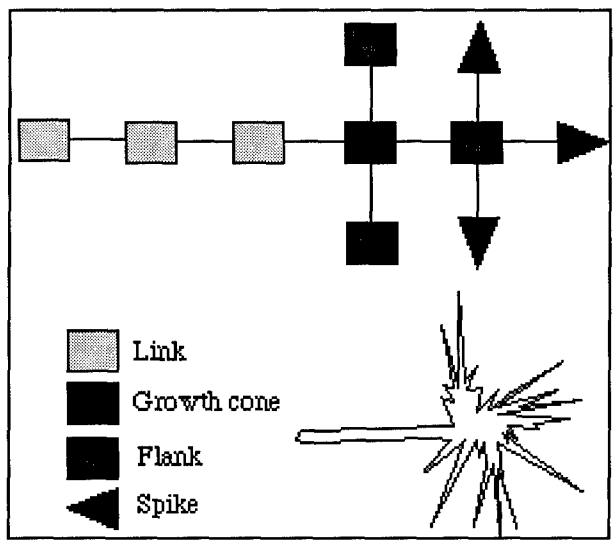


Figure 5: The different axon-element states.

When the development of the agent morphology has settled into a configuration in which divisions no longer occur (but the cell state can still continue to evolve) a control architecture or 'nervous system' develops on top of the arrangement of cells. This happens because specialized cells, i.e. those that express a specific protein (*tAxon*), will send out axons that will innervate other cells, and as such establish a neural network architecture. Only cells expressing the *tTarget* protein will be innervated, and axons will only grow on top of cells that express *tCAM* proteins. This is modeled directly after one postulated mechanism in biological development, in which neurons emit growth cones that can detect the presence of certain molecules (Cellular Adhesion Molecules or CAMs) on the surface of cells and adjust their direction of growth accordingly.

The central feature of this model is the 'growth cone' model, illustrated in figure 5. The black rectangle represents a growth cone. It is linked back to the cell from which the axon originated by 'link' elements, and it sends out 'flanks' that sample the neighborhood ahead by means of 'spikes'. The flank whose spikes detect more *tCAM* proteins will be promoted to a growth cone in the next time step, with the axon splitting in different directions in case of a tie. This fairly intricate finite state model is modeled on the workings of a real biological growth cone, albeit quite simplified.

After the process of axon growth is complete, a dynamical neural network (Beer and Gallagher 1992) is instantiated that connects sensor, interneuron and actuator cells according to the connections made during the neural developmental phase. Time constants and biases of the organism are global and are specified separately in the genome.

Thus, emergence and placement of sensor and actuator cells needs to be coordinated with appropriate neural developmental events to lead to interesting behavior. Sensor cells that did not send out an axon or actuators that were not innervated will have no effect on the agents' behavior.

A Braitenberg Hate Vehicle

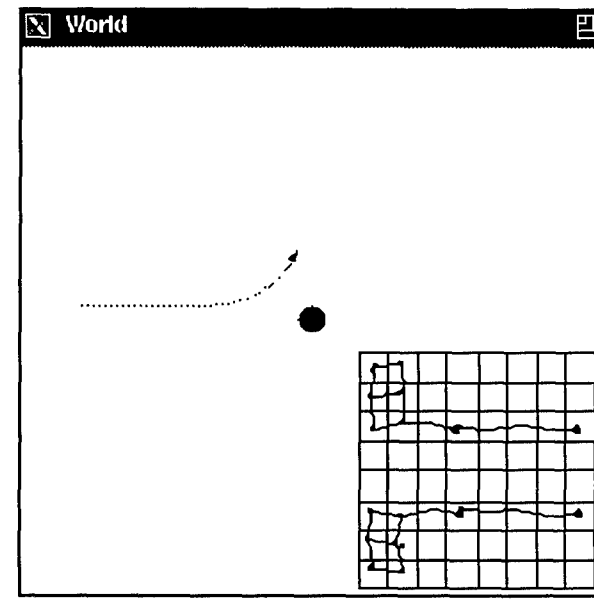


Figure 6: The behavior of the hand-designed Braitenberg Hate Vehicle in a simulated world.

In order to explore the expressiveness of our developmental model, we have hand-designed a genome capable of directing the development of both the body and nervous system of a simple Braitenberg-style "Hate Vehicle". This agent executes a simple avoidance task in a simulated world. Figure 6 shows the adult form of the hand-designed organism and its behavior on the task; sensors are at the frontal side of the organism (on the right in the figure), and a simple network relays their activation to patches of actuator cells (on the left). Figure 7 shows the expression domains of the different

proteins in the final developmental stage of the organism. A complete analysis of the developmental sequence is beyond the scope of this paper, but can be found in (Dellaert 1995).

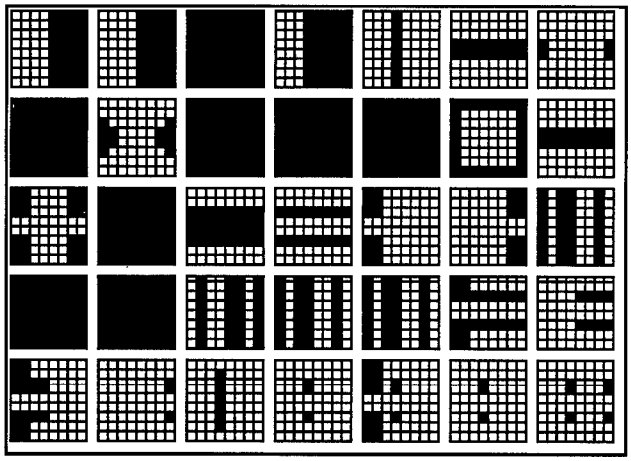


Figure 7: The different 'protein expression domains' in the adult organism. Each square in the matrix shows in which cells each of the 35 proteins is present. For example, the bottom right inset where the *tAxon* protein is expressed, and corresponds to the cells in figure 6 that send out an axon.

It is a demonstration of the power of our model that the initial genetic specification can direct the simultaneous development of both morphology and nervous system, leading to a *complete* autonomous agent. We want to stress that only the genome has been manually specified (i.e. a set of fully specified operons) and that all subsequent development follows from the model without intervention. Note also that no learning takes place in the agent. All the behavior it exhibits is solely a function of its evolved architecture.

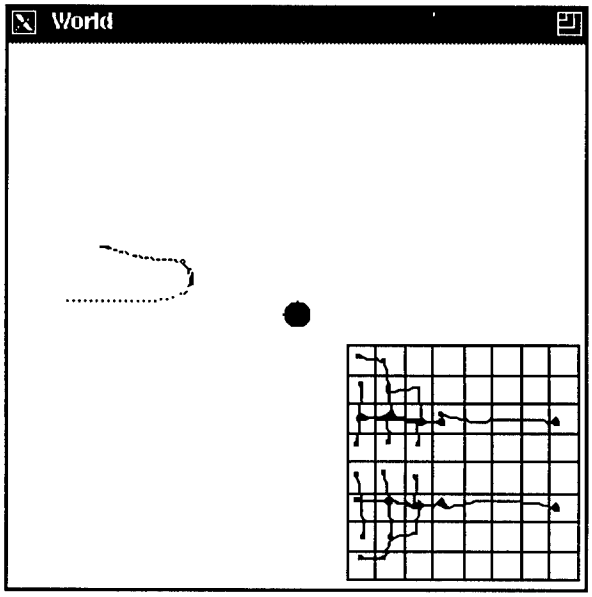


Figure 8: The behavior of the incrementally evolved agent in a simulated world.

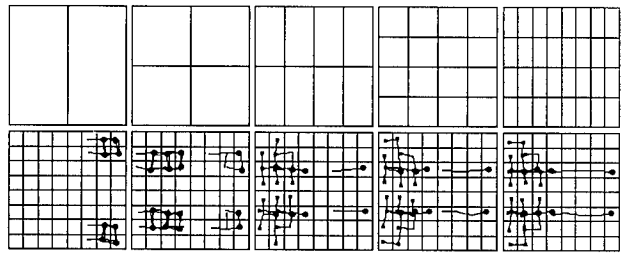


Figure 9: The consecutive steps in the development of the incrementally evolved agent. See text for an explanation.

Although we were unable to evolve such an agent from scratch, we were able to significantly improve its performance through incremental evolution. We started out with the hand-designed genome as a primer for the genetic algorithm, and used a performance function to evaluate each agent's aptness for the avoidance task. The result was an agent that was markedly better at executing the task, as shown in Figure 8. The consecutive steps in the development of this incrementally evolved agent are shown in figure 9. In this figure, the big black dots represent axon-emitting cells, whereas the smaller black squares represent innervation of target cells. Time goes from top left to bottom right, the last square representing the adult organism.

Lessons Learned

Alas, the expressiveness of the developmental model as illustrated in the previous paragraphs is also its downfall. There is a vast space of possible genomes with their associated phenotypes. Many of the protein expression domains need to be tightly coordinated with one another to have a working nervous system emerge. Thus, both the size and structure of the search space make the problem hard. Although the Braitenberg example convinces us that viable organisms (with respect to the task) exist, it does not give us a path to them starting from random genotypes. Although we *were* able to incrementally evolve better performing agents starting with the hand-designed genome (see also Dellaert and Beer 1994b), we have not been able to obtain convincing results when starting evolution from scratch. In addition, the sheer complexity of the model makes it quite hard and error-prone to implement. To cope with both these problems, we have experimented with a drastically simplified model.

4. A Simplified Model

The simplified model uses a random Boolean network (RBN) as an abstraction for the genome, where the state of each cell is equal to the state of its RBN. The topology and rules of the RBN are the same for each cell and can be regarded as the genetic specification of the organism. The model for neural development has been simplified considerably, and is now based on the range and position of the interacting cells. In this section we will discuss each of these aspects in more detail.

The Random Boolean Network Model

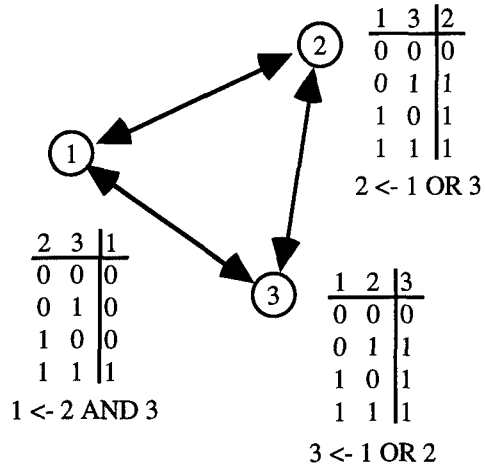


Figure 10: A simple RBN. Boolean functions are specified as truth tables. Example from (Kauffman 1993).

When we first started this work, we used a simpler model than the genome-cytoplasm model sketched above, based on random Boolean networks. RBNs were first thought of as an abstraction for genetic regulatory networks by (Kauffman 1969; Kauffman 1993) and extended by (Jackson, Johnson and Nash 1986) to systems of multiple, communicating networks as needed in the context of development.

A random Boolean network can be represented by a graph, for example the simple RBN of figure 10. In an N node network, each node is defined by K incoming edges, defining a pseudo-neighborhood, and a particular Boolean function[2]. The edges can be recurrent, i.e. nodes can connect to themselves. In addition, each node has an associated state variable assuming a value of 0 or 1. Each node synchronously updates its state in discrete time steps, and the state of a node at time t+1 is the value of the Boolean function using the state of the input nodes at time t.

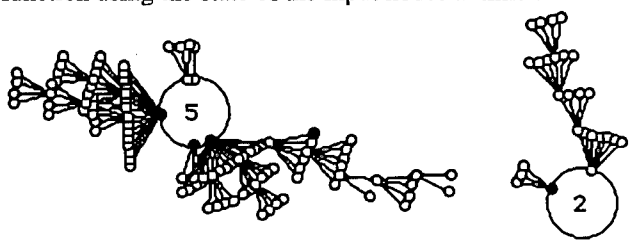


Figure 11: The phase portrait of a random Boolean network with N=7 and K=3. See text for an explanation.

An RBN can then be used as an abstraction of the genetic regulatory network inside a cell. The state of the cell can be equated to the state of the RBN, and each node can be seen as equivalent to a particular 'protein', although we will not use that terminology here. The topology and the particular Boolean functions that specify the RBN will then determine how the state evolves over time, and thus correspond to the 'genome' of the complex model. The incoming edges for each node indicate which other nodes influence its state, and can be seen to correspond to the input tags of an operon in the earlier model. If we complement this with a mechanism for intercellular communication, allowing for edges to occur between cells, we have a very similar picture to the more complex model sketched above.

One nice additional property of RBNs, however, is that their dynamics can be made explicit using phase portraits (Wuensche 1993; Wuensche 1994). This provides us with a powerful tool to analyze the dynamics of development (Dellaert 1995). An example is shown in figure 11. In the figure, each small circle represents one of the 128 states of the 7-node RBN, and the edges between them represent state transitions. The transitions always occur in the direction of the state attractors, in this case state cycles respectively with period 5 and 2.

Development

The developmental simulation unfolds similarly as with the more complex model. Each cell contains an identical copy of the RBN, but the state of the RBN can vary between cells. Instead of a given protein signaling division, a cell will now divide if a prespecified bit in the state vector is set. This bit is set in the 'egg cell' to ensure at least one division event. Symmetry breaking occurs by deterministically perturbing the state (by flipping one bit) of one of the daughter cells resulting from this first division. Intercellular communication is accomplished by calculating neighborhood state vectors that serve as external inputs to the intracell RBNs, perturbing their phase portrait. Details can be found in (Dellaert and Beer 1994a); the final result is a developmental sequence that is qualitatively similar to the ones obtained by the more complex model, but at a considerably reduced computational cost.

Simplified Neural Developmental

To cope with the problems that arose when using the complex model described above, we have implemented a much simplified neural developmental model.

As in the earlier model, the final differentiation of a cell determines whether it will send out an axon or not, and/or whether it is a target for innervation. This is done simply by associating a prespecified node of the RBN with these respective properties. In addition, three bits in the RBN state vector determine whether the cell will be a sensor, an actuator or an interneuron, and one bit specifies whether any innervation will be inhibitory or excitatory.

The development of the neural network is simple and straightforward: each cell that has the 'axon bit' set will innervate all target cells (with the 'target bit' set) within its range. The constants specifying the connection strength and the range at which cells innervate each other are evolved together with the genome, and are identical for all cells.

Thus, unlike before, the developmental process does not need to specify elaborate pathways of CAM molecules on which the axons can grow, but merely needs to make sure

[2]The 'random' refers to the fact that each node can have a different Boolean function and pseudo-neighborhood, in contrast to cellular automata which represent a special case of RBNs.

that it places cells that need to be connected within each other's range. In addition, it can adjust the architecture by specifying for each cell individually the sign of the connection and the cell type. The range and weight factors can be co-evolved with the placement of the cells to obtain the desired behavior.

A Line Follower

Using this simpler model, we have not only evolved agents that execute the earlier avoidance task quite well, but have also succeeded in evolving agents that perform more difficult tasks, in this case following a line made up of circular segments.

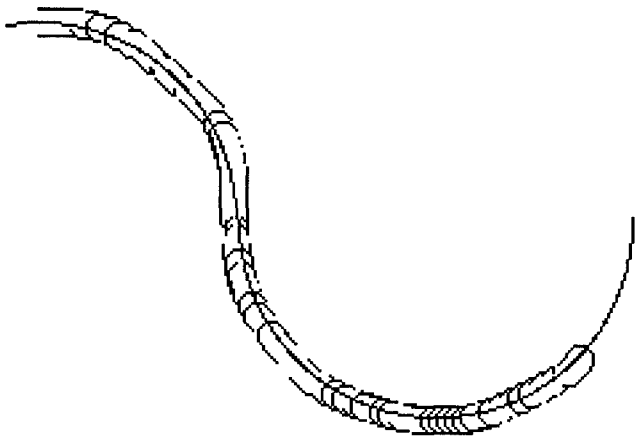


Figure 12: The outline of the line following agent as it executes the task. The agent moves from left to right.

In this experiment, each agent in a genetic algorithm population (Steady state GA with tournament selection, population size 100, mutation rate 2.5%, crossover rate 100%, tournament size 7, run for 10000 evaluations) was put into a simulated world and evaluated on how well it could follow the curve sketched in figure 12, made up of two semicircles. In the neural developmental phase, standard static neurons were used instead of the dynamical neurons of before.

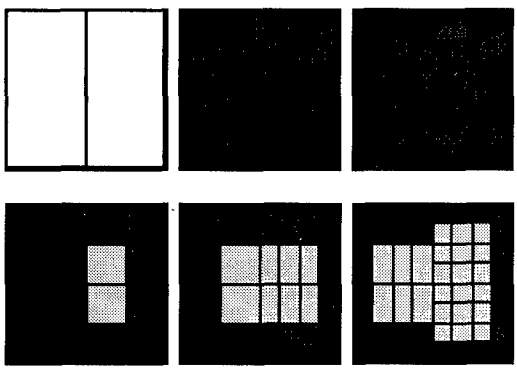


Figure 13: The developmental sequence, from the two cell stage onwards, of the line-following agent.

The evaluation function used was the integrated squared error to an ideal position along the curve. This can be easily calculated since the velocity of each agent is kept constant: only the steering angle varies. The response of each sensor cell in the agent is a Gaussian function of its closest distance to the line. The output of the actuators is averaged for each side, added together and multiplied by a constant factor to become a steering output. In the following paragraph we describe the structure of one such agent, the best of one particular run of the GA, whose behavior in the simulated world is shown in figure 12.

The developmental sequence of the evolved agent is shown in figure 13. In this figure, the dark cells represent sensors while the lighter ones are the actuators. As you can see, development unfolds asymmetrically as not all cells divide an equal number of times. Also, you can see that the sensor cell type is being induced by the perimeter (as shown before in figure 4). No interneuron type cells were present.

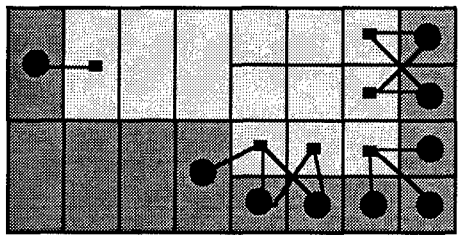


Figure 14: The innervation of actuators by sensors in detail. Only the bottom half of the agent is shown.

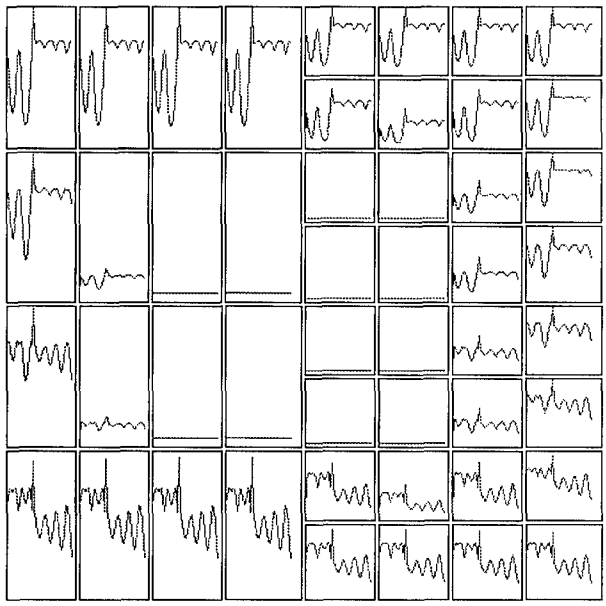


Figure 15: The activation of each neuron/cell during the execution of the task. Time is on the x-axis of each cell, the bottom and top of the cell represent an activation of 0 and 1

If we look at the detailed innervation of the cells, we get the picture sketched in figure 14, where all the connections turned out to be excitatory. This makes sense when you consider how movement is implemented. By this innervation, the agent steers in the direction where it can sense the line. Note that this particular innervation depends on the range factor: with a bigger innervation range more cells would be

innervated. With this picture in mind, it is easy to understand figure 15, which shows the activation of each neuron/cell during the execution of the task. You can see that the actuators not innervated by any sensors remain inactive throughout the whole task.

When evaluating the evolved line-followers on novel lines of different curvatures, their control strategy turned out to be unstable. They were able to follow the line for a while, but they eventually overshot and lost track of the line. Given that they have not been presented with different types of lines during the time that they were evolved it is reasonable that they experience these problems. We are investigating whether we can evolve dynamically stable controllers by exposing them to many different line-following tasks during evolution.

5. Discussion

In this paper we have tried to incorporate a developmental model in the artificial evolution of *complete* autonomous agents, i.e. with a 'body' *and* a 'nervous system'. We presented two models that operated at different levels of abstraction from the biological phenomenon of development, although we feel that both possess many of the properties that are at the core of development, most notably the dynamic interplay between genome and cell state. In addition, they are both able, albeit in quite different ways, to account for the emergence of a nervous system on top of a developed multicellular body arrangement.

The first and most complex model is more biologically defensible in its details. It represents an initial attempt at extending our earlier work (Dellaert and Beer 1994a) to include neural development. We were able to demonstrate its expressiveness by showing that it can account for the development of agents complete with a neural control architecture, capable of executing a simple avoidance task in simulation. We were also able to incrementally evolve better agents starting from that hand-designed agent.

The second and simplified model was adopted in the hope that we could use it to evolve agents from scratch. It used a model for the genome and cell state that was somewhat more removed from the biological example, random Boolean networks. These have the advantage of being computationally cheaper and they lend themselves to analysis using tools from dynamical systems theory. In addition, the second model used a considerably simplified model for neural development. We have used it successfully to evolve agents from scratch that can execute simple tasks. The line-following agent was presented as an example.

We believe that both models can be useful in future research. Both can conceivably be used in the quest to understand more about the logic of development. In the case of the RBN model (and possibly also for the genome-cytoplasm model), it is possible to use analysis tools like phase portraits to visualize what is going on during the simulated developmental process. We might one day be able to use the abstractions that are explored here to visualize the dynamical trajectories being traversed by cells in actual biological development. In addition, we can try and understand more by trying to synthesize observable aspects of development using these models. Some of this has already been explored in (Dellaert 1995). We have, among other things, examined the use of phase portraits to visualize the dynamics of interconnected RBNs (as used in our model for intercellular communication), and we have tried to synthesize pattern formation mechanisms like those found underlying the development of compound insect eyes.

In the domain of autonomous agents, we would like to see whether more complex tasks are within reach of the model. A first simple extension would be to evolve line followers with a stable controller for any type of line, given some smoothness constraint. We have also experimented with introducing learning into the process. If an agent is capable of learning during its lifetime, we might benefit from the Baldwin effect to speed up evolution (Whitley, Gordon and Mathias 1994). Finally, it would be of interest to investigate the behavior of the model under selection of an implicit fitness function, i.e. where agents are placed in a world in which their only task is to outsmart their peers. Here, we expect the complexity of the simulated world to be reflected in the complexity of the agents, and perhaps this brings out the potential of a developmental model better than using explicitly designed fitness functions.

Development is an important, powerful and integral element of biological evolution. It is our hope that explorations such as those we have presented here will contribute to its understanding, both in its own right and as an element of autonomous agent research.

Acknowledgments

Special thanks to James Thomas, Katrien Hemelsoet, and Shumeet Baluja for their helpful comments. This work was supported in part by grant N00014-90-J-1545 from the Office of Naval Research.

References

Beer, R.D. and J.C. Gallagher. 1992. Evolving dynamical neural networks for adaptive behavior. *Adaptive Behavior* 1:91-122.

Belew, R.K. 1993. Interposing an Ontogenic Model between Genetic Algorithms and Neural Networks. In *Advances in Neural Information Processing Systems (NIPS) 5*, edited by Hanson, S.J., J.D. Cowan, andC.L. Giles. Morgan Kauffman: San Mateo.

Cangelosi, A., D. Parisi, and S. Nolfi. 1993. Cell Division and Migration in a 'Genotype' for Neural Networks, Technical PCIA-93, Institute of Psychology, C.N.R.-Rome.

de Boer, M.J.M., F.D. Fracchia, and P. Prusinkiewicz. 1992. Analysis and Simulation of the Development of Cellular Layers. In *Artificial Life II*, edited by Langton, C.G., *et al.* Addison-Wesley: Reading, MA.

De Garis, H. 1994. CAM-Brain: the genetic programming of an artificial brain which grows/evolves at electronic speeds in a cellular automata machine. In *Proceedings of the First IEEE Conference on Evolutionary Computation*, IEEE: New York.

Dellaert, F. 1995. Toward a Biologically Defensible Model of Development, Masters Thesis, Case Western Reserve University.

Dellaert, F. and R.D. Beer. 1994a. Toward an Evolvable Model of Development for Autonomous Agent Synthesis. In *Artificial Life IV, Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems*, edited by Brooks, R. and P. Maes. MIT Press: Cambridge, MA.

Dellaert, F. and R.D. Beer. 1994b. Co-evolving Body and Brain in Autonomous Agents using a Developmental Model, Technical Report CES-94-16, Dept. of Computer Engineering and Science, Case Western Reserve University.

Fleischer, K. and A.H. Barr. 1994. A Simulation Testbed for the Study of Multicellular Development: The Multiple Mechanisms of Morphogenesis. In *Artificial Life III*, edited by Langton, C.G. Addison-Wesley: Reading, MA.

Gruau, F. and D. Whitley. 1993. The cellular development of neural networks: the interaction of learning and evolution, Research 93-04, Laboratoire de l'Informatique du Parallélisme, Ecole Normale Supérieure de Lyon.

Jackson, E.R., D. Johnson, and W.G. Nash. 1986. Gene Networks in Development. *J. theor. Biol.* 379-396.

Jakobi, N. 1995. Harnessing Morphogenesis, Technical Report School of Cognitive and Computing Sciences, University of Sussex.

Kauffman, S. 1969. Metabolic Stability and Epigenesis in Randomly Constructed Genetic Nets. *J. theor. Biol.* 437-467.

Kauffman, S.A. 1993. *The Origins of Order.* Oxford University Press: New York.

Kitano, H. 1994. Evolution of Metabolism for Morphogenesis. In *Artificial Life IV, Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems*, edited by Brooks, R. and P. Maes. MIT Press: Cambridge, MA.

Kodjabachian, J. and J.-A. Meyer. 1994. Development, Learning and Evolution in Animats. In *Proceedings of PerAc '94: From Perception to Action, Lausanne*, edited by Gaussier, P.;.N., J.-D. IEEE Comput. Soc. Press: Los Alamitos, CA.

Lyndenmayer, A. and P. Prusinkiewicz. 1989. Developmental Models of Multicellular Organisms: A Computer Graphics Perspective. In *Artificial Life*, edited by Langton, C.G. Addison-Wesley: Reading, MA.

Mjolsness, E., D.H. Sharp, and J. Reinitz. 1991. A Connectionist Model of Development. *J. theor. Biol.* 429-453.

Nolfi, S., O. Miglino, and D. Parisi. 1994. Phenotypic Plasticity in Evolving Neural Networks. In *First Conference From Perception to Action*, edited by Lausanne (pp.

Sims, K. 1994. Evolving 3D Morphology and Behavior by Competition. In *Artificial Life IV, Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems*, edited by Brooks, R. and P. Maes. MIT Press: Cambridge, MA.

Whitley, D., V.S. Gordon, and K. Mathias. 1994. Lamarckian evolution, the Baldwin effect and function optimization. In *Parallel Problem Solving from Nature - PPSN III*, edited by Davidor, Y., H.-P. Schwefel, andR. Manner. Springer-Verlag: Berlin, Germany.

Wilson, S.W. 1989. The Genetic Algorithm and Simulated Evolution. In *Artificial Life*, edited by Langton, C.G. Addison-Wesley: Reading, MA.

Wuensche, A. 1993. Memory, Far from Equilibrium. In *Proceedings, Self-Organization and Life: From Simple Rules to Global Complexity, European Conference on Artificial Life (ECAL-93)*, edited by Brussels, Belgium (pp. 1150-1159).

Wuensche, A. 1994. The Ghost in the Machine: Basins of Attraction of Random Boolean Networks. In *Artificial Life III*, edited by Langton, C.G. Addison-Wesley: Reading, MA.

Evolution of Plastic Neurocontrollers for Situated Agents

Dario Floreano
Laboratory of Cognitive Technology
AREA Science Park
Trieste, Italy
dario@psicoiris.area.trieste.it

Francesco Mondada
Laboratory of Microcomputing
Swiss Federal Institute of Technology
Lausanne, Switzerland
mondada@di.epfl.ch

Abstract

In this paper we investigate a novel approach to the evolutionary development of autonomous situated agents based on the assumption that the neural mechanisms underlying ontogenetic learning are themselves developed and shaped by the evolutionary process. A genetic algorithm is used to evolve neural structures that can be continuously modified during life according to the mechanisms specified in the genotype. The evolutionary process is carried out on a real mobile robot. The analysis of one of the best evolved individuals shows rapid development of stable behavior mediated by fast-changing synapses which are dynamically stable.

1 Introduction

Adaptation to the environment takes place at multiple levels and time-scales: it ranges from the long-term dynamics of *phylogenetic evolution* to the fast process of *ontogenetic learning*. At the neural level, these adaptation modalities, that are well-timed and co-ordinated, are responsible for the development of robust and complex control systems that display the ability to self-regulate their own behavior and keep the organism alive.

Behaviors which are inherited at birth and cannot be modified by experience are called *innate behaviors* or *instincts*[1] [11]. To this extent, most of the experiments on *animats* which involve some form of artificial evolution have been concerned with the evolution of innate behaviors. However, there are a few experiments which combine learning and evolution making use of carefully designed architectures and traditional supervised learning algorithms (e.g., see [13, 1, 5]).

In this paper we investigate a different approach based on the assumption that the neural mechanisms underlying ontogenetic learning are themselves developed and shaped by the evolutionary process (see section 5 for biological considerations). A genetic algorithm is used to evolve neural structures that can be continuously modified during life according to the mechanisms specified in the genotype. Each decoded network is downloaded into a mobile robot which is let free to interact with the environment while its fitness is automatically computed and stored away for selective reproduction.

[1] The term *instinct* has undergone several re-definitions which have stressed the influence of experience and maturation on the final behavior. Here we assume the definition given by Darwin, whereby instincts are the product of natural selection and inheritance.

2 Experimental setup and task description

The experimental setup employed in these experiments is identical to that already described in [4], which we summarize below. We used the miniature mobile robot Khepera [12] which has a circular shape with a diameter of 55 mm, a height of 30 mm, and a weight of 70 g. Khepera is supported by two wheels and two small Teflon balls. The two wheels are controlled by two DC motors equipped with incremental encoders (12 pulses per mm of advancement of the robot) and can move in both directions. Each motor controller sets the speed of its own wheel according to a continuous value between -0.5 and +0.5, where 0.0 means no rotation, -0.5 means maximum rotation speed in one direction (set to 80 mm/s) and 0.5 means maximum rotation speed in the opposite direction. Each of the eight Infra-red proximity sensors, six positioned facing one direction of motion and two the opposite direction, returns a continuous value between 0 and 1 that signals the distance of an object from that sensor (the closer the object, the higher the value returned). In our environment the maximum detection range was approximately 4 cm. The robot was provided with a small positioning device which detected light beams emitted by a laser device placed on the top of the environment and computed the robot absolute position (see [5] for further details). This information was used only for behavior analysis and was not passed to the neural controller. Khepera was attached via a serial port to a Sun SparcStation by means of a lightweight aerial cable and specially designed rotating contacts.

The robot was put in an environment consisting of a

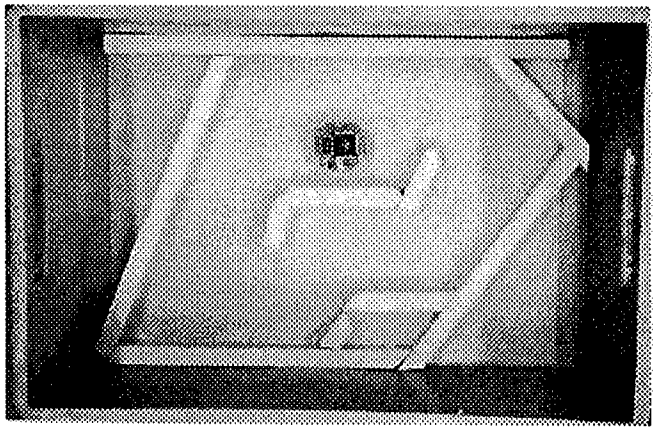


Figure 1: Bird-view of the environment with the robot.

circular corridor whose external size was approx. 60x50 cm large (Figure 1). The walls were made of light-blue polystyrene and the floor was made of gray paperboard. The robot could sense the walls with the IR proximity sensors. The environment was within a portable box positioned in a room always illuminated from above by a 60-watt bulb light.

The genetic operators, the decoding routines from genotypes to phenotypes, and the neural network dynamics were managed by the workstation CPU. Each individual of a population was in turn decoded into the corresponding neural network, the input nodes connected to the robot sensors, the output nodes to the motors (output unit activation was transformed in the range ± 0.5 before passing it to the motor), and the robot was let free to move for 24 s (80 motor actions) while its fitness Φ was automatically recorded and accumulated. Each sensory-motor loop lasted 300 ms (to/from communications between the robot and the workstation lasted approximately 60 ms) during which the wheel speed was kept constant. Between one individual and the next, a pair of random velocities was applied to the wheels for 5 seconds: this procedure was aimed at limiting the artifactual inheritance of particular locations between adjacent individuals in the population.

The fitness function Φ was designed to evolve obstacle avoidance and straight navigation behaviors, as in [4]

$$\Phi = V\left(1-\sqrt{\Delta v}\right)(1-i) \qquad (1)$$

$$\begin{aligned} 0 &\leq V &&\leq 1 \\ 0 &\leq \Delta v &&\leq 1 \\ 0 &\leq i &&\leq 1 \end{aligned}$$

where V is a measure of the average rotation speed in absolute value of the two wheels, $\Delta v = (v_{left} + 0.5) - (v_{right} + 0.5)$ is the absolute value of the difference between the speed of the wheels transformed into positive values, and i is the activation value of the proximity sensor with the highest activity. Φ was newly computed every 300 ms, accumulated during the life of the individual, and finally normalized by the number of actions. Since the robot has a circular shape and the wheels can rotate in both directions, this function has a symmetric surface with two equal maxima, each corresponding to one motion direction.

3 Genetic Encoding and Network Dynamics

A simple genetic algorithm (as described in [6]) with linear fitness scaling, roulette-wheel selection, one-point crossover, and bit-substitution mutation was employed (see details in the Appendix) to evolve binary chromosomes which encoded a set of parameters describing synapses properties and learning rules (see section 5 for motivations). Every time a phenotype was created, its synapses were intialized to small random values and could change their strength during life; final strengths were *not* coded back into the chromosome. Thus, each decoded neural network changed its own synaptic strength configuration according to the genetic instructions and without external supervision while the robot interacted with its own environment.

Each neural network had only three neurons — one hidden neuron and two motor neurons, each receiving synaptic connections from all the eight IR sensors and from the hidden neuron (Figure 2); this architecture could not be modified in the experiments described here.

Synaptic connections could have a *driving* or a *modulatory* effect on the postsynaptic neuron; afferent signals were combined in a two-component activation function [14] which gave an output between 0 and 1 (Figure 3). Driving signals determined whether the unit activity was below or above 0.5 (which, when transformed into the range ± 0.5 for motor control, was the point of inversion of wheel rotation), whereas modulatory signals could enhance or dampen the unit response, but could not change the direction of wheel rotation.

Synapses were individually coded on the chromosome. Each synapse was described by a set of four properties: whether it is driving or modulatory (1 bit), whether it is excitatory or inhibitory (1 bit), its learning rule (2 bits), and its learning rate (2 bits).

Each individual synapse could change its strength according to one of four basic Hebbian learning rules: pure Hebbian, postsynaptic, presynaptic, and covariance (see [20]). We have slightly modified each of these rules in order to satisfy the following constraint. Synaptic strength cannot grow indefinitely, but is intrinsically bound in the range $[0.0, 1.0]$ by means of a self-limiting mechanism which depends on the current synaptic strength; this so-

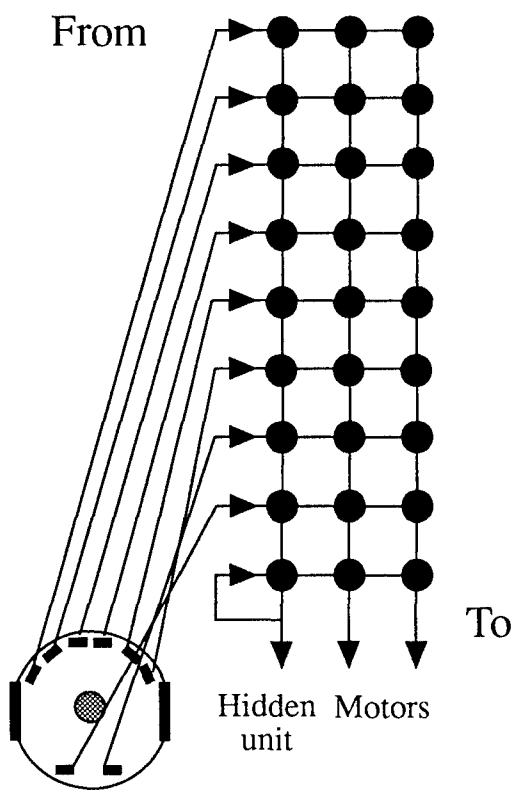


Figure 2: The architecture of the neural network employed. The black circles are the synapses; the circle in the middle of the robot body represents the hidden unit. The activations of the three units correspond — respectively — to the hidden units, the left motor, and the right motor.

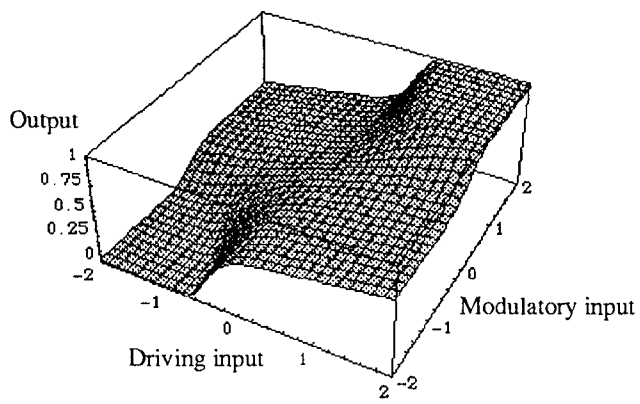


Figure 3: The activation function of internal nodes and motor nodes [14]: signals coming from driving connections and modulatory connections are separately integrated and passed to the network.

lution has the property of keeping the sign of the synapse unchanged, thus reducing the degrees of freedom of the network and putting more emphasis on the genetically evolved configuration of excitation and inhibition. The four types of synaptic change Δw are as follows (where x, y are respectively the pre- and postsynaptic activation and range between 0.0 and 1.0):

- The simplest learning mechanism is plain Hebb, whereby synapses can only be strengthened

$$\Delta w = (1 - w)\, xy \tag{2}$$

- The postsynaptic rule is similar to the plain Hebbian rule, but also decreases the synaptic efficacy when the postsynaptic unit is active and the presynaptic unit is not

$$\Delta w = w\,(-1 + x)\, y + (1 - w)\, xy \tag{3}$$

- Instead, in the presynaptic learning rule the decrement takes place when the presynaptic unit is active, but the postsynaptic unit is inactive

$$\Delta w = wx\,(-1 + y) + (1 - w)\, xy \tag{4}$$

- The covariance rule here takes the form of a synchronous- activation detector: if the presynaptic and postsynaptic activity levels differ by more than half the maximum node activation, the synaptic efficacy is reduced in proportion to that difference, otherwise it is increased in proportion to the difference

$$\Delta w = \begin{cases} (1 - w)\mathcal{F}(x, y) & \text{if } \mathcal{F}(x, y) > 0 \\ (w)\mathcal{F}(x, y) & \text{otherwise} \end{cases} \tag{5}$$

where $\mathcal{F}(x, y) = \tanh(4(1 - |x - y|) - 2)$ is a measure of the difference between the presynaptic and postsynaptic activity. $\mathcal{F}(x, y) > 0$ if the difference is bigger or equal to 0.5 (half the maximum node activation) and $\mathcal{F}(x, y) < 0$ if the difference is smaller than 0.5.

As soon as the network is decoded and attached to the sensors and motors of the robot, synaptic weight values are intialized to small random values in the range $[0.0, 0.1]$ and are updated every 300 ms according to the following discrete-time equation

$$w_t = w_{t-1} + \eta \Delta w_t \tag{6}$$

where η is the learning rate, which can assume one of four values $\{0.0, 0.3, 0.7, 1.0\}$. If the learning rate is 0.0, that synapse will not change its strength during the life of the individual.

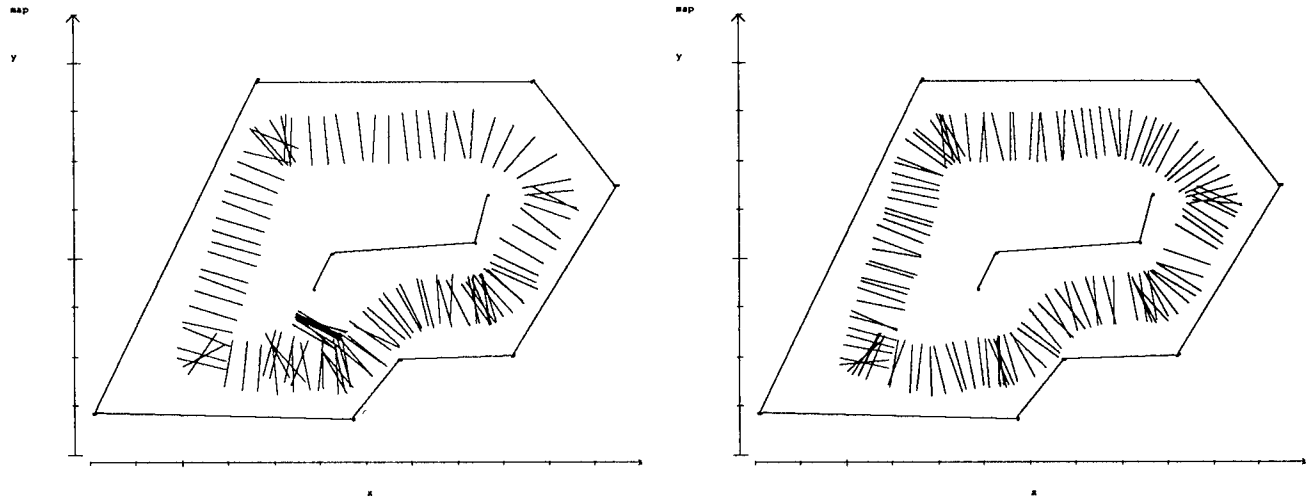


Figure 4: Trajectory of the robot that learns to navigate during life. Position data, visualized as bars representing the axis connecting the two wheels, were acquired with the laser positioning system every 100 ms. Data refer to the best individual of the last generation of one evolutionary run. Left: trajectory during the first lap (the robot starts in the lower portion of the environment and turns anti-clockwise). Right: trajectory at the second lap.

4 Results

Three different runs of this experiment were made. In all cases the best individual fitness reached a maximum value around the 50th generation ($\Phi = 0.23, \pm 0.09$). When compared to the results of the experiment reported in [4] where we evolved only synaptic strengths for the same task of obstacle avoidance and straight navigation, the fitness values recorded here displayed higher variation.

All the best neural networks of the last generation could control the robot in order to keep a straight trajectory while avoiding obstacles. The evolved behaviors resulted in smooth paths around the arena (Figure 4). This ability was developed by each individual neurocontroller during the first few sensory-motor loops, whatever the inital random values assigned to the synapses. In all the three runs the best individuals of the last generation moved in the direction where more IR sensors were placed, which provided a better sampling of the obstacles facing the robot.

The evolved neurocontrollers varied in the type of behavioral strategies and learning modalities both within a single population and across the three evolutionary runs. Here we present an analysis of the best individual of the last generation of one run and in section 5 we shall give some data on how this individual differed from the others.

The neural network was decoded, connected to the robot sensors and motors, the synaptic strengths were intialized to random values in the range $[0.0, 0.1]$, the robot was positioned facing a corner of the inner wall (Figure 4, left; initial position corresponds to the set of superimposed bars in the lower portion of the environment) and let free to move. During the first 2 s (6-7 synaptic updates) the robot adjusts its position alternating backward and forward motions until it finds a wall on its right side. This initial behavior is quite stereotypical: it is displayed for any starting position. Once the wall is found, the robot moves forward keeping the wall at a distance of 2 cm from its own right side; every second or third action, it slightly turns toward the wall and then continues on the previous direction. This sort of jerky behavior is gradually reduced when moving along a straight long wall (e.g., along the north and east walls). If the wall is slightly bent, the robot can still follow it without reducing speed, but when the walls form a convex angle smaller than 90 degrees (which means that most of the front IR sensors are active) the robot stops, backs and rotates to the right, and then moves forward again in the new direction. The robot has developed a sort of wall-following strategy. After one lap of the corridor, the path becomes smoother with less trajectory adjustements and more tuned to the geometric outline of the environment (Figure 4, right).

4.1 Internal Dynamics

The development of such behavior can be understood if one looks at the internal dynamics of the evolved network. Figure 5 plots the strengths of all the synapses in the network during the first 100 actions (sensory-motor loops) visualized in Figure 4 where the plots are laid out in the same format as the synapses in Figure 2. The

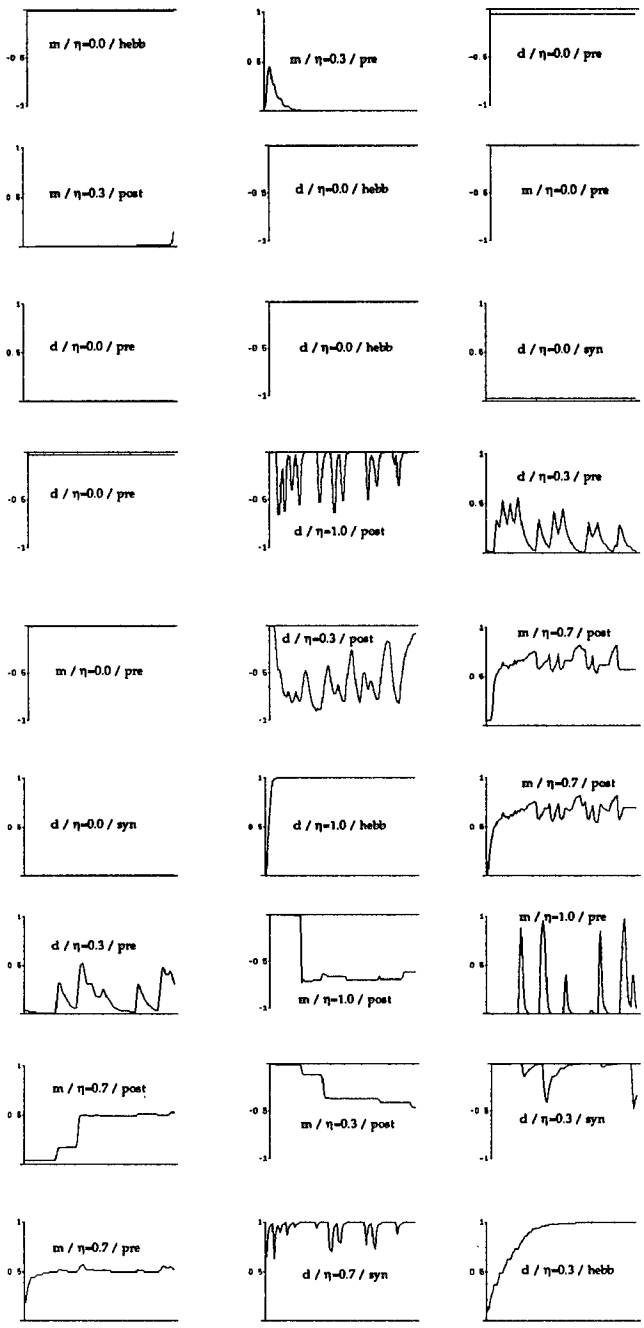


Figure 5: Synaptic strength recorded every 300 ms during the first 100 actions of the robot. The synapse layout is as depicted in Figure 2: rows represent input from the sensors (clockwise starting from the leftmost front sensor) and the hidden unit, columns represent input to the hidden unit, the left motor neuron and the right motor neuron. Details of the synapse characteristics are given in the following order: role (d = driving; m = modulatory), learning rate (η), learning rule (hebb = pure hebb; post = postsynaptic; pre = presynaptic; syn = synchrony). The sign of the y-axis indicates whether the synapse is excitatory or inhibitory.

neural network does not use the internal node to process the sensory information because most of the afferent synapses from the front sensors have the learning rate set to 0.0 (the presynaptic rule applied to the synapse from the second front sensor keeps its strength to 0.0 in normal conditions, i.e. when the robot follows a wall to its own right, but it can increase its efficacy when something is detected on the left). However, the internal node is maintained moderately active most of the time by excitatory connections from rear sensors and excitatory self-connection.

The behavioral preference for following walls on the right side is reflected by the null efficacy (for both wheels) of the afferent synapses from sensors on the left side: the only adjustable connection — from the leftmost sensor to the left wheel — uses a postsynaptic mechanism that enables only temporary excitation of the wheel and thus obstacle avoidance if something happens to be on the left side. Synapses from the internal node to the motor neurons are both driving and excitatory: given the constant level of activation of the internal node, these synapses provide a constant forward motion to both wheels. All the remaining synapses to the motor neuron that controls the right wheel are excitatory: that means that the right wheel will only move forward, its speed mainly generated by the internal node but modulated by the information coming from the two rightmost sensors. The excitatory driving synapse from the central sensor to the right wheel causes fast accelerations and decelerations which, combined with the opposite effect of the corresponding synapse to the other motor neuron, cause the backward rotations when something is frontally detected. Synapses connecting the three sensors on the front-right side to the left wheel are very important (they all convey driving signals): they are responsible for the wall-following behavior. The synapse corresponding to the rightmost sensor (nearly always highly active) quickly learns to transmit constant and high excitation to the left wheel. This excitation — that would otherwise lead the robot against the wall — is counterbalanced by the driving inhibitory inputs that come from the other two front-right sensors: the corresponding synapses display an oscillatory pattern that is responsible for the frequent small turns toward the wall. These turns are important because provide information about the wall curvature by allowing all the three right-front sensors to receive information on distance from the wall. The fast changing synapse from the front sensor provides rapid temporary inhibition to the left wheel when some object is frontally detected or the walls form a sharp convex angle smaller than 90 degrees; its pattern of strength change is in phase with that of the excitatory synapse from the same sensor to the right wheel: this causes the robot to stop and turn backward to re-adjust its trajectory. As the robot gradually adapts to the geometry of the environment,

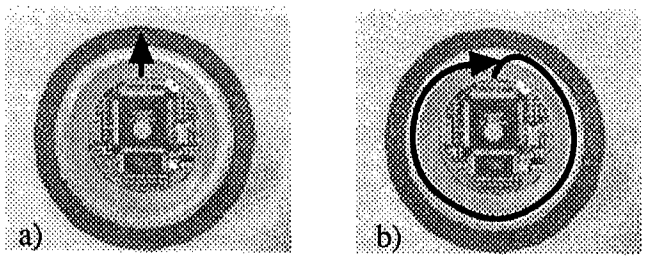


Figure 6: Sensory deprivation: a) the robot starts its life within the fence; b) the robot is put within the fence after the development of wall-following.

these turns become less frequent, which is reflected by the decreasing frequency and intensity of change of this synapse (and of the corresponding synapse to the other wheel).

4.2 Environment, Inherited Structures, and Learning

We have performed a number of tests on the same individual described above in order to investigate the influence of the training environment, and the relationships between inherited structure, early experiences, and learning. In a first test we have put the robot in an empty space and initialized the synaptic strengths with random values. After the initial short backward motion, the observed behavior was rather erratic, mostly composed of long forward motions and rapid turns in place to the left. If the robot was then put in the training environment, it would soon acquire the typical wall-following behavior. When put again in the empty space, it would start to alternate right turns (searching for the wall that suddenly disappeared) with sudden stops.

To better understand the role of early experience, we tried to produce learning in conditions of "sensory deprivation" by putting the robot within a small circular fence with an internal diameter of 75 mm (Figure 6). In this condition, all the sensors have roughly the same high activation, without regard to the action taken by the robot. When we put the robot at birth within the fence, it started to push in the frontal direction against the fence and it did not change behavior. Instead, when we put it within the fence after the usual learning phase in its normal environment, it began pushing frontally for one second or two, and then it started to turn slowly to the right still slightly pushing against the fence. The latter behavior, which is more varied and useful (it could help to find an opening –if there was one), could be achieved only if the phase of sensory-motor co-ordination had been properly completed in the normal environment.

In order to understand the relationship between the inherited neural structure and learning, we put the robot in the training environment, initialized its synaptic values to small random values, and disabled learning on all synapses. Typically, the robot would move straight and crash into the nearest wall without being able to recover, although in some cases it could manage to perform a left turn at the correct place. When we put this learning-impaired individual in an empty area, we observed that it performed long straight trajectories (approximately 40 cm) in the frontal direction interrupted by sharp right turns in place. Although this behavior does not allow proper navigation in a cluttered environment, nonetheless the trajectory roughly reflects the geometry of the environment where the robot has been evolved. These tests indicate that inherited structure does have a few basic and primitive skills (going forward and turning to the right) that have been shaped by the environment and the selection criterion. These inherited abilities, or "instincts", narrow the search space of learning providing a good starting point for a fast development of the ontogenetically acquired behavior.

5 Discussion

The genetic alphabet employed in these experiments was motivated by a set of computational considerations and neurophysiological findings. There is no special reason to believe that synaptic plasticity in biological systems can be explained in terms of a single learning mechanism; rather, individual synapses might modify their own strength according to different learning rules. The choice of a particular learning rule depends upon the types of receptors and transmitters found at the synaptic locus. Biologists have recently isolated a number of genes regulating the expression of NMDA receptors [9] which are thought to be the most likely mechanism responsible for Hebbian learning. On the other hand, neurophysiologists have provided evidence for the existence of a few types of Hebbian-like rules, where synapses are coincidence detectors that increase or decrease their own efficacy depending on the simultaneous activity level of the presynaptic and postsynaptic cells [10, 19, 16, 18]. Thus, it might be argued that the learning properties of synapses are specified in the genetic material, just like any other characteristics of the nervous system. Rather than coding chemical properties of synapses and simulating molecular dynamics, we simply coded four simple Hebbian rules for which there is neurophysiological evidence. These rules were modified by including a normalizing factor dependent on current synaptic efficacy that constrained maximum synaptic strength, as reported in [22], and could not change the sign of the synapse; this self-limiting mechanism had the computational advantage of avoiding the risk of saturating the activation function and thus reduced the search space of ontogenetic learning. Also the choice of encoding synaptic sign (which could not be in-

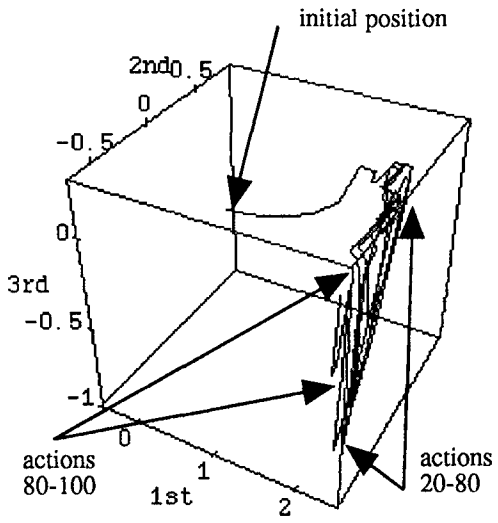


Figure 7: State-space representation of synapse dynamics during the first 100 actions plotted as trajectory within the space of the first three principal components. Arrows indicate the starting position and the range of oscillation between action sequences 20-80 and 80-100. Oscillations within the subspace of the third (smallest) component correspond to trajectory adjustments. Method: Sanger's network [15] for extracting the first three principal components of the input correlation matrix was trained to stability on the 27-component vectors corresponding to the synaptic activity recorded during the first 100 actions of the robot (visualized in Figure 5); after training, input vectors were presented again to the network and output unit activations were plotted.

verted by learning, as in biological nervous systems) was aimed at reducing the search space of ontogenetic learning and at putting more emphasis on inherited wiring of the neural structure. The choice of a two-component activation function combining driving and modulatory signals was motivated only by biological considerations; modulatory synapses, which are widespread in biological nervous systems and are modifiable (e.g., see [8]), are traditionally considered responsible for delivering contextual information [17, 3]. Here, we wanted to see how they could be exploited for sensory-motor control (note that when modulatory signals are absent, the activation function is equivalent to a standard sigmoid function).

Our results indicate that it is possible to evolve learning structures with emergent fast-adaptation properties. The evolved individual analyzed above displays interesting properties. The stable behavior acquired during life is regulated by continuously changing synapses which are *dynamically stable*. In the conventional view, synapses are relatively stable and slow components of the nervous system. Synaptic changes are identified with the learning of new skills or acquisition of new knowledge, while neuron activations are identified with the expression of behavior and of existing knowledge.[2] Typically, acquisition of a stable behavior in a static environment corresponds to stability (no further change) of individual synapses (e.g., see [7]). Such requirement is explicitly included into the objectives (least-mean-square error minimization, energy reduction, maximization of node mutual information, etc.) from which — both supervised and unsupervised — conventional learning algorithms are derived, but it is not included into the fitness function employed here, which is defined only in behavioral terms. The functioning of our system offers a complementary — but not necessarily alternative — explanation. Synapses are responsible for both learning and behavior regulation. Knowledge in the network is not expressed by a final stable state of the synaptic configuration, but rather by a dynamic equilibrium point in a n-dimensional state-space (where n is the number of synapses). Figure 7 plots the trajectory of synaptic change in the reduced state-space of the first three principal components of the recorded synaptic vectors during the first 100 actions of the individual displayed in Figure 4. During the first 6 actions the systems moves toward a subregion of the space for which there is no change in the first two principal components; residual variation along the slice of space corresponding to the third principal component corresponds to trajectory adjustments and is further reduced as the robot gradually tunes its path to the geometry of the environment.

In the case of the the neurocontroller analyzed above, most of the active synapses are excitatory (75%), but inhibition plays a key role in controlling some crucial aspects of the behavior, such as active sampling of the wall curvature and trajectory re-adjustment. Although there is not a preference for a particular learning rule, the evolutionary procedure has made sparse use of the plain Hebbian mechanism. This "choice" is quite reasonable because that rule does not allow reduction of synaptic efficacy and may thus hamper future adaptation. Where the Hebb rule has been employed, the links between the units (rightmost sensor to right and left wheel and internal node to right wheel) correspond to the establishment of a basic and "immutable" aspect of behavior, i.e. forward motion. The network employs the internal node as a sort of internal pattern generator[3] [2] that drives the organism forward even when there is not sensory stim-

[2]This view has been recently challenged by Yamauchi and Beer [21], who have evolved and analyzed continuous-time recurrent neural networks that give the external appearance of performing reinforcement learning while, in fact, these networks have fixed connection weights and use only internal node dynamics.

[3]The analogy should not be taken literally because the network dynamics and the use of wheels, rather than of legs, do not necessarily require the pattern of activity found in living organisms.

ulation available. It is interesting to note that internally generated forward motion is regulated by modulatory synapses from the two rightmost sensors to the right wheel: signals coming from these synapses contribute to adjust the distance of the robot from the wall.

The behavioral tests described in section 4.2 indicate that the structure of training environment plays a great role in shaping both the inherited neural structures and the properties of the ontogenetically developed behavior. Whatever the environment where the robot is placed, the neural network actively seeks an object and then performs a set of actions aimed at keeping it on its own right side. Its adaptation abilities are limited only to those variations that were encountered in the training environment (e.g., walls with different curvatures) and this is the reason for the development of a simple wall-following behavior. To this extent, such a plastic system evolved in stationary environments does not offer significant adaptation advantages w.r.t. systems with fixed synaptic weights. Our current research is focused on testing the procedure here described to changing environments.

A final remark concerns the robustness of the method employed. The relatively high variation in fitness values across runs (in the range ± 0.05 to ± 0.11) reflects a loose correspondence between the genotype specification and the phenotype performance: this means that a small change in the chromosome of the individual might result in drastic (and potentially maladaptive) changes at the phenotypic level. This is also reflected in the diversity –within a population and across runs– of the evolved neural structures. Although continuously changing synapses, dominance of excitatory vs. inhibitory synapses, and sparse use of plain Hebb learning are a common feature of the best 5-7 individuals in the last generation, all the remaining individuals display different structures and maladaptive behaviors. This variation can be partly explained by intrinsic variation of a not-yet converged population, but we think that it might be further reduced by employing more suitable building blocks (currently under investigation).

Appendix

Genetic algorithm parameters:

Population size	80
Generation number	50
Chromosome length	162 bits
Crossover probability	0.1
Mutation (expected probability of each bit being flipped)	0.2
Life length	80 actions
Action duration	300 ms

Acknowledgments

We would like to thank Edo Franzi and André Guignard for important work in the design of Khepera, and Randall Beer for useful comments on an early draft of this paper. Dario Floreano and Francesco Mondada have been partially supported by the Swiss National Research Foundation (project PNR23).

References

[1] D. H. Ackley and M. L. Littman. Interactions between learning and evolution. In C.G. Langton, J.D. Farmer, S. Rasmussen, and C. Taylor, editors, *Artificial Life II: Proceedings Volume of Santa Fe Conference*, volume XI. Addison Wesley: series of the Santa Fe Institute Studies in the Sciences of Complexities, 1991.

[2] F. Delcomyn. Neural basis of rhythmic behavior in animals. *Science*, 210:492–498, 1980.

[3] A. K. Engel, P. Koenig, A. K. Kreiter, T. B. Schillen, and W. Singer. Temporal coding in the visual cortex: new vistas on integration in the nervous system. *Trends in Neuroscience*, 15:218-226, 1992.

[4] D. Floreano and F. Mondada. Automatic creation of an autonomous agent: Genetic evolution of a neural-network driven robot. In D. Cliff, P. Husbands, J. Meyer, and S. W. Wilson, editors, *From Animals to Animats III: Proceedings of the Third International Conference on Simulation of Adaptive Behavior*. MIT Press-Bradford Books, Cambridge, MA, 1994.

[5] D. Floreano and F. Mondada. Evolution of homing navigation in a real mobile robot. *IEEE Transactions on Systems, Man, and Cybernetics-Part B*, 26:in press, 1996.

[6] D. E. Goldberg. *Genetic algorithms in search, optimization and machine learning.* Addison-Wesley, Reading, MA, 1989.

[7] J. Hertz, A. Krogh, and R. G. Palmer. *Introduction to the theory of neural computation.* Addison-Wesley, Reedwood City, CA, 1991.

[8] J. A. Hirsch and C. D. Gilbert. Long-term changes in synaptic strength along specific intrinsic pathways in the cat visual cortex. *Journal of Physiology*, 461:247–262, 1993.

[9] M. Hollman and S. Heinemann. Cloned glutamate receptors. *Annual Review of Neuroscience*, 17:31–108, 1993.

[10] S.R. Kelso, A.H. Ganong, and T.H. Brown. Hebbian synapses in hippocampus. *Proceedings of the National Academy of Sciences USA*, 83:5326–5330, 1986.

[11] D. J. McFarland. *The Oxford Companion to Animal Behaviour.* Oxford University Press, Oxford, 1981.

[12] F. Mondada, E. Franzi, and P. Ienne. Mobile robot miniaturization: A tool for investigation in control algorithms. In *Proceedings of the Third International Symposium on Experimental Robotics*, Kyoto, Japan, 1993.

[13] D. Parisi, F. Cecconi, and S. Nolfi. Econets: Neural networks that learn in an environment. *Network*, 1:149–168, 1990.

[14] W. A. Phillips, J. Kay, and D. Smyth. The discovery of structure by multi-stream networks of local processors with contextual guidance. *Network*, 6:225–246, 1995.

[15] T. D. Sanger. Optimal unsupervised learning in a single-layer feedforward neural network. *Neural Networks*, 2:459–473, 1989.

[16] W. Singer. Activity-dependant self-organisation of synaptic connections as a substrate of learning. In J. P. Changeux and M. Konishi, editors, *The Neural and Molecular Bases of Learning.* Wiley, London, 1987.

[17] W. Singer. Search for coherence: a basic principle of cortical self-organization. *Concepts in Neuroscience*, 1:1–26, 1990.

[18] P. K. Stanton and T. J. Sejnowski. Associative long-term depression in the hippocampus induced by hebbian covariance. *Nature*, 339:215–218, 1989.

[19] G.S. Stent. A physiological mechanism for hebb's postulate of learning. *Proceedings of the National Academy of Sciences USA*, 70:997–1001, 1973.

[20] D. Willshaw and P. Dayan. Optimal plasticity from matrix memories: What goes up must come down. *Neural Computation*, 2:85–93, 1990.

[21] B. Yamauchi and R. D. Beer. Sequential behavior and learning in evolved dynamical neural networks. *Adaptive Behavior*, 2(3):219–246, 1995.

[22] X-D. Yang and D. S. Faber. Initial synaptic efficacy influences induction and expression of long-term changes in trasmission. *Proceedings of the National Academy of Science USA*, 88:4299–4303, 1991.

Increasing Adaptivity through Evolution Strategies

Ralf Salomon
AI Lab, Department of Computer Science, University of Zurich
Winterthurerstrasse 190, 8057 Zurich, Switzerland
FAX: +41-1-363 00 35; Email: salomon@ifi.unizh.ch

Abstract

The automated generation of controllers for real-world autonomous agents by means of evolutionary methods has recently attracted a lot of attention. Most of the pertinent research has employed genetic algorithms or variations thereof. We have applied a different evolutionary method to the generation of a control architecture for Braitenberg vehicles, namely "evolutionary strategies". The application of the ES accelerates the development of such controllers by more than one order of magnitude (a few hours compared to more than two days). This result is very important, since the development process is to be done in real systems. In addition to the dramatic speedup, there is an important theoretical reason for preferring evolutionary strategy over genetic algorithms, namely epistatic interaction.

1 Introduction

Autonomous agents are self-sufficient, embodied systems (robots) [5] that do some useful work. Autonomous means that the agent operates without any human control. Self sufficient means that the agent can maintain its internal energy level over a long time; typically, the agent gets some reward after doing useful work. Autonomous agents are equipped with sensors and effectors, such as motors or grippers. An agent perceives its environment through sensors like infrared sensors and it manipulates its environment by actively using the effectors. One major goal of research in autonomous agents is to study intelligence as the result of a system environment interaction, rather than understanding intelligence on a computational level. Thus, autonomous agents are very important in the field of new AI. For an overview of special issues involved in autonomous agent design see, for example, [16, 25].

The control structures of autonomous agents can be developed either by the designer or by applying evolutionary algorithms. Many research projects [6, 8, 9, 14, 18] have used genetic algorithms (GAs) to evolve neural control structures for autonomous agents. As pointed out in Harvey et al. [14] the automated evolution is more appropriate, since it allows the development of complex control structures that exhibit many interactions between sub-systems. In this paper, we too advocate the use of evolutionary algorithms for adaptive systems.

Even though GAs have been successfully applied to rather small tasks, an increasing amount of research gives strong evidence that GAs are very time consuming if the parameters exhibit epistatis. Epistatis describes the interaction of parameters with respect to the fitness of an individual. Results of other research [22, 23] strongly indicate that the independence of parameters is an essential prerequisite of the GA's high global convergence performance; the presence of epistatis drastically slows down convergence. The problem in the context of autonomous agents is that the parameters of control systems for real-world applications are not independent. Consequently, even the evolution of rather simple controllers, such as Braitenberg vehicles, is very time consuming. To relieve this problem, it is suggested [14] to divide the evolution of more complex systems into different stages.

Until now, most research projects have focused on GAs, which are only one option of evolutionary algorithms. Surprisingly little attention has been devoted to the evolution strategy (ES) as defined by Rechenberg [20] and Schwefel [24]. The ES is especially designed for applications that involve real-valued parameters. To this end Section 2 gives a brief comparison of these two algorithms with respect to autonomous agents. One result of this comparison is that the performance of the ES does not degrade in the presence of epistatis; the ES behaves invariant with respect to epistatis.

The theoretical analysis of Section 2 is supported by a case study on the evolution of Braitenberg vehicles. To this end, Section 3 describes the setup of the case study. The case study consists of the *Khepera*^{TM} robot, which is an example of a small robot that is widely used for research on autonomous agents. Subsection 3.1 describes Khepera in more detail. In order to achieve the goal of operating autonomously, an agent has to perform different tasks, such as exploring the environment, moving around, avoiding obstacles and so forth. The agent's behavior is controlled by a control system, which is typically implemented as a neural network. Such a control system uses the sensor readings and its internal state to

determine the agent's next action. Subsection 3.2 describes a simple neural control architecture called Braitenberg vehicle [4], and Section 3 describes the arena, in which the robots have to operate.

Section 4 demonstrates the benefits of the ES when applied to the development of adaptive control architectures. That section reports some experiments obtained by applying a (3,6)-ES to the evolution of Braitenberg vehicles. A (3,6)-ES evolves reasonable Braitenberg controllers within 30 generations, which takes approximately one and a half hours. Other research [8, 9, 18] use GAs for almost the same task. However, the GA-based approach requires approximately two and a half days. Thus, the ES speeds up the developmental process by more than one order of magnitude. This speed up is of great practical relevance, since the experiments are to be done in real systems that dynamically interact with the real world. Also, such a time reduction allows for the application to more complex control structures, which are currently untracktable.

One important feature of GAs is their ability to use genomes with variable length. Section 5 briefly shows how this important feature can be incorporated in the ES. Finally, Section 6 concluded with a summary of the work presented in this paper.

2 Evolutionary Algorithms

Evolutionary algorithms can be seen as a framework that includes genetic algorithms, evolutionary strategies, evolutionary programming, and genetic programming. All these algorithms are heuristic population-based search procedure based on natural selection and population genetics. Typically, such population-based search procedure generate offspring in each generation. Then a fitness value (defined by a fitness function) is assigned to each offspring. Depending on the fitness, each population member survives with a certain probability.

All evolutionary algorithms implement different strategies, and therefore, each algorithm is best suited for a different application domains. To this end, the following subsections give a brief description of GAs and ES. A good comparison of evolutionary algorithms can be found in [2]. Section 2.3 compares both schemes in more detail with respect to autonomous agents, adaptivity, and performance issues under epistatis. Furthermore, both algorithms are well founded by an extensive theory, which can be found in, for example, [3, 12, 17, 24].

2.1 Genetic Algorithm

The GA technique is based on Holland [13] and a good introduction to GAs can be found in [12]. There does not exist *one* GA, but rather a variety of variants, each covering different applications and aspects. According to [24], the canonical GA can be described as follows:

Step 0: Initialization of the population's individuals and evaluation of the individuals' fitness
Step 1: Selection of the parents according to a preselected selection scheme (e.g., roulette wheel, linear ranking, truncation selection)
Step 2: Recombination of selected parents by exchanging parts of their genes
Step 3: Mutation of some genes by a prespecified probability
Step 4: Go to Step 1

Before using a GA in a particular application domain, the designer is concerned with the coding of the parameters, the implementation of the mutation and recombination/crossover operators, and the choice of the selection scheme. Traditionally, a GA treats every parameter as a bit string. Depending on the required precision, a coding scheme can encode each parameter by a certain number of bits and an attached mapping function. The mapping function maps the bit representation to real-valued parameters. Another coding scheme is to directly use the bit representation used by the programming language. In such a case, no mapping function is needed. Other algorithms like the BGA [17] treats each parameter as a real-valued data type and implements mutation by adding or subtracting small random numbers.

In each generation, a GA typically draws pairs of parents and applies mutation and recombination with probabilities p_m and p_r respectively. After generating a specified number of offspring, the GA evaluates each offspring and selects the best population members as parents for the next generation. Typical selection schemes are roulette wheel selection, linear ranking, or truncation selection. Very often, GAs use an elitist selection scheme, which preserves the best individual in order to maintain gained success. The influence of different selection schemes on the resulting convergence can be found in, for example, [26].

When using GAs, it is very important to find appropriate parameter settings. Normally, the mutation probability p_m is set to small values $p_m \approx 1/n$ [2, 7, 12, 17, 19], where n denotes the number of parameters, e.g., the number of weights in a neural network controller. Generally, GAs prefer rather high recombination rates [2, 12, 17]. If using one-point or two-point crossover, the recombination probability p_r is set to values between 0.5 and 0.9, and if using uniform recombination, the probability is set to $p_r = 0.5$.

This GA framework has been successfully applied in various domains, such as function optimization, VLSI design, neural network learning, and autonomous agent development. Applications of GAs in the field of autonomous agents can be found in, for example, [6, 8, 9, 14, 18].

2.2 Evolution Strategy

The ES has been introduced in [20] (see, also, [24]). The ES is similar to GAs. A $(\mu \dagger \lambda)$-ES maintains a population of μ parents and generates λ offspring in each generation. In a (μ, λ)-ES, the μ parents are selected from best λ offspring, whereas in a $(\mu + \lambda)$-ES, the μ parents are selected from the union of parents and offspring. For further details see, for example, [2]. Currently [2], the (μ, λ)-ES is recommended, especially in noisy environments.

In contrast to GAs, the ES encodes each parameter as floating-point numbers, and it applies mutation to *all* parameters simultaneously, i.e., $p_m = 1$. Mutations are typically implemented by adding $(0, \sigma)$-normal distributed random numbers. The key concept of the ES is that it, in its simplest form, maintains one global step size σ for each individual. This step size is self-adapted by the following mechanism. Each offspring inherits its step size from its parent(s). This step size is modified by log-normal random numbers prior to mutation. By this means, the step size is self-adapted to nearly optimal values, since, in a statistical sense, those offspring survive that have the best adapted step size. For further details of different step size schemes see [2, 24]. In addition, the ES can use the same recombination schemes as GAs [2].

The self-adaptation mechanism of the step size σ has a great advantage. It allows the ES to self-adapt to different fitness landscapes. Therefore, besides the population size, the ES does not have any parameters that have to be tuned by the designer. Since the ES directly encodes each parameters as floating-point numbers, the ES is better suited for problems, such as neural networks, that are specified by a set of parameters.

Evolutionary programming (EP) has been introduced by Fogel [10]. EP is another evolutionary algorithm and very similar to ES. Recent applications to general function optimization can be found in [11]. Since both schemes are very similar, it can be expected that EP yields very similar results when applied to the evolution of Braitenberg vehicles.

2.3 Genetic Algorithm vs. Evolution Strategy

Even though the differences between GAs and ES seem rather small, they significantly influence the performance of both algorithms and, consequently, they aim at different problem domains. The remainder of this subsection discusses some important aspects.

The main differences between both algorithms are that GAs apply mutations to only a few parameters per offspring, whereas the ES applies mutation to *all* parameters, i.e., $p_m = 1$. Furthermore, the ES encodes each parameter as floating-point number, whereas GAs have, in the general case, to worry about the coding scheme. Traditionally, GAs encode each parameters as bitstrings. However many application that involve real-valued parameters [17, 18], directly use floating-point numbers. In addition, the ES features a self-adaptation mechanism, which self-adapts the step size by itself so that no parameter settings have to be done by the user.

From the coding mechanisms it should become clear that ES is rather suited for real-valued parameters, whereas GAs with traditional bit coding schemes are preferred for combinatorical tasks like the traveling-salesman problem. Furthermore, the ES increases adaptation, since the *algorithm is inherently adaptive*.

Many applications [1, 2, 7, 17]. report high performance when applied to various optimization task, especially the optimization of multimodal function that contain millions of local optima but only one global optimum. The high performance that was achieved in these applications suggest that GAs easily escape from local optima and that they have very good global convergence. However, recent results [22] show that the performance of GAs drastically degrades, if a rotation is applied to the coordinate system. Furthermore, theoretical analysis [23] shows that the computational complexity of GAs can increases up to $O(n^n)$, when applied to multimodal functions with n parameters that depend on each other. Even when applied to simple unimodal functions, epistatis drastically slows down the convergence of GAs [22]. Moreover, that analysis suggests that the independence of the parameters is an essential prerequisite for a high performance of GAs.

A rotation of the coordinate system does not change the fitness function (fitness landscape), but it induces *epistatis*. Epistatis describes the interaction of different parameters with respect to the fitness function. In other words, if epistatis between parameters is present all parameters involved have to be adapted simultaneously in order to achieve any improvement of the fitness function; adapting only one parameter, leads to a worse fitness. Thus, applying mutation to only one parameter is not sufficient in such situations.

In a particular application, one important question is, whether the parameters are independent or if they depend on each other, i.e., whether epistatis is present. When looking at the control structures of autonomous agents, the controller's parameters are not independent in the general case. To this end, Section 4 investigates the evolution of Braitenberg controllers. It turns out that the weights of the Braitenberg network are not independent. The results show that even a simple (3,6)-ES is more that one order of magnitude faster than GA-based approaches.

It could be argued that an acceleration of the development of controllers by one order of magnitude is not that important. However, such an improvement is of great practical interest. First of all, control architectures that allow the agent to adapt to the environment have

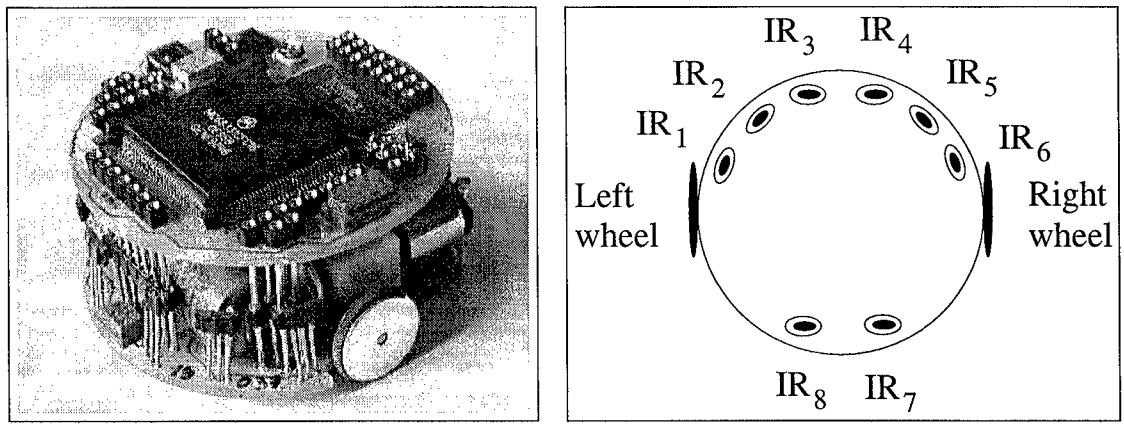


Figure 1: The Khepera robot. The right part shows the approximate location of the infrared sensors $IR_1 \ldots IR_8$.

to be developed in real hardware. Thus, it is important whether the developmental process consumes only a few hours or if it take several days. Second, a moderate convergence speed becomes more important as the complexity of the control architectures increases. From a theoretical point of view, any randomized algorithm eventually finds a solution with a probability strictly greater than zero. However, from a practical point of view, time matters. Thus, good convergence speed allows for the development of more complex control architectures, which, in turn, allows the agent to better adapt to its environment and the tasks, which are to be done by the agent. In summary, the results presented in Section 4 strongly suggest that the research community should devote more attention to the evolution strategy or evolutionary programming.

3 The Experimental Setup

This section describes the experimental setup for the evolution and optimization of Braitenberg vehicles. First, Subsection 3.1 describes the Khepera robot, which is widely-used in research on autonomous agents. Then, a Braitenberg architecture is described in Subsection 3.2, and, finally, Subsection 3.3 describes the arena, in which the robot has to operate.

3.1 The Khepera Robot

The Khepera robot (cf. Fig. 1) is 55 mm in diameter and 32 mm high. The robot is equipped with eight infrared and eight ambient light sensors as well as two motors, which can be controlled independently. Khepera's sensors and motors are controlled by a Motorola 68331 micro controller. Khepera can operate in two modes. In the first mode, a program is downloaded into the onboard memory, which allows Khepera to operate without any further hardware. In the second mode, Khepera is connected with a workstation via a serial link. In the experiments reported in this paper, the robot was controlled from a workstation and the ambient light sensors were not used. A detailed description of the robot and its electrical parts can be found in [15].

The motors can be controlled independently of each other by sending commands (i.e., function calls) to the robot. Valid speed values are in the range $[-40, 40]$; inside the program, these values are normalized such that they are in the range $[-1, 1]$. To avoid problems caused by the floating-point-to-integer conversion, [0,1]-equal-distributed random numbers are added to the motor speeds at each time step. By setting both motors to the same speed but with different signs, for example, the robot spins on the spot. The robot interprets the speed settings as commands. Internal PID controllers take care of the robot's dynamics. However, rapidly changing motor commands induce additional dynamics, which can cause problems for the fitness evaluation. By means of attached wheel encoders, the robot measures the motor's real speed, which differ from the command setting in situations, in which the robot cannot move. The real speeds can be obtained by sending special commands to the robot.

Khepera is equipped with eight infrared proximity sensors. The sensor readings are of type integer and the values are in the range $[0, 1023]$. Within the program, the sensor readings are normalized such that the values are in the range $[0, 1]$. The sensors give reasonable input values for object distances between 10 mm and 60 mm. A sensor value of 1023 indicates that the robot is very close to the object, and a sensor value of 0 indicates that the robot does not receive any reflection of the infrared signal.

A major problem with Khepera is that the sensor readings are very noisy, which causes several problems for the fitness evaluation of a given controller, i.e., the weights of the Braitenberg network. The effect of the noisy sensors to the fitness evaluation can be seen in Fig. 2. Figure 2 shows how the fitness contributions f_t are dynamically changing under constant environmental conditions,

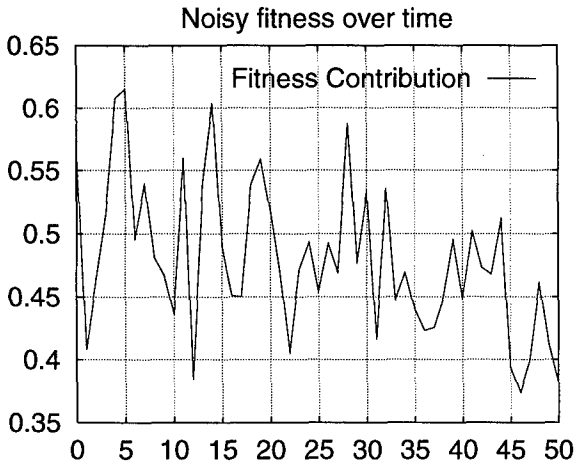


Figure 2: The noisy sensor readings cause a dynamical change of the individual fitness contributions f_t even under constant environmental conditions.

i.e., constant motor speeds, constant ambient light, and constant position in the arena. Furthermore, the sensor readings are subject to several environmental conditions, such as the material the object is made of, the object's surface, and the ambient light. During a long series of experiments, the environment cannot be kept constant. In summary, the fitness evaluation is extremely noisy.

3.2 *Braitenberg Vehicles*

As outlined in the introduction, an autonomous agent cannot sit somewhere while doing nothing, since it would consume energy and would eventually die. Rather, the agent has to *operate* in its environment. Thus, moving around while avoiding obstacles is a key issue in autonomous agent research. Braitenberg [4] has proposed a simple architectures for such tasks. Figure 3 shows a control architecture inspired by a Braitenberg type-3c vehicle. The main idea is that a sensor with a high proximity activation accelerates the motor on the sensor's side whereas this sensor slows down the motor on the opposite side. By this principle, the presence of an obstacle leads to different motor speeds, which causes the robot a turn. Depending on the activation of all proximity sensors, the robot either turns, spins on the spot, or even backs up.

Braitenberg type-3c architectures are simple and straight forward, but finding appropriate weights is everything but easy. The control architecture is typically implemented as a neural network. The activation of the left motor M^l is calculated by the following formula

$$M^l = \sum_{i=1}^{8} IP_i w_i^l + w_0^l \quad , \tag{1}$$

where IP_i denotes the activation of the proximity sensor i and w_i^l denotes the weight that connects proximity

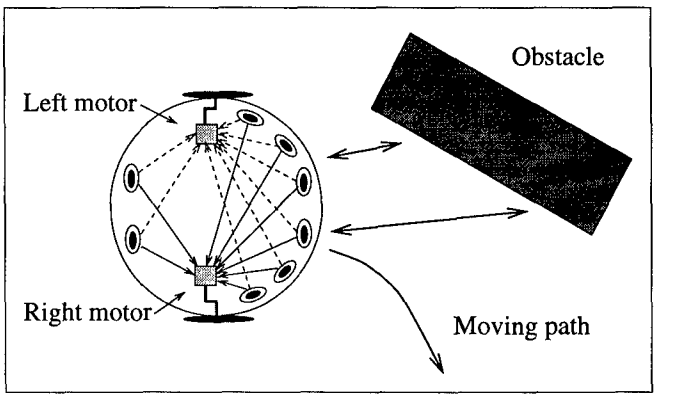


Figure 3: A control architecture inspired by a Braitenberg type-3C vehicle. The sensory information controls the motors via inhibitory and excitatory connections.

sensor IP_i with the left motor. The weight w_0^l represents the idle activation of the left motor. This "bias" weight is responsible for the robot's forward moving in the absence of any obstacle. The calculation of the right motor's activation is similarly given by

$$M^r = \sum_{i=1}^{8} IP_i w_i^r + w_0^r \quad . \tag{2}$$

The main problem is to determine the weights w_i^l, w_i^r, w_0^l, and w_0^r such that the robot is moving around while it avoids obstacles. Mostly, this is done in a trial and error process. Standard neural network training procedures cannot be used, since it is not reasonable to determine a set of training patterns prior to training. Thus, the evolution strategy is an ideal candidate for this optimization problem.

3.3 *The Experimental Setup*

The experimental setup has been chosen as close as possible to setups proposed in other research [8, 9, 18]. Figure 4 shows the arena, in which the robot has to move. The arena is of size 60x45 cm and the walls are made from wood with a height of 3 cm. The width of the corridors is chosen such that always at least one proximity sensor has a less-than-maximum value.

As already discussed, the robot in such an arena has to move forward quickly while it has to avoid obstacles. In order to evolve good Braitenberg vehicles, the fitness function has to incorporate the motor speeds and the distance to obstacles. However, using speed and distance only is not sufficient. In such a case, a robot that is spinning on the spot with a high speed far away from any obstacle would have a high fitness. But such a robot would not do anything useful. Therefore, the fitness function is to be enhanced by a third term that favors straight movements by penalizing turns (see also [8, 9, 18]).

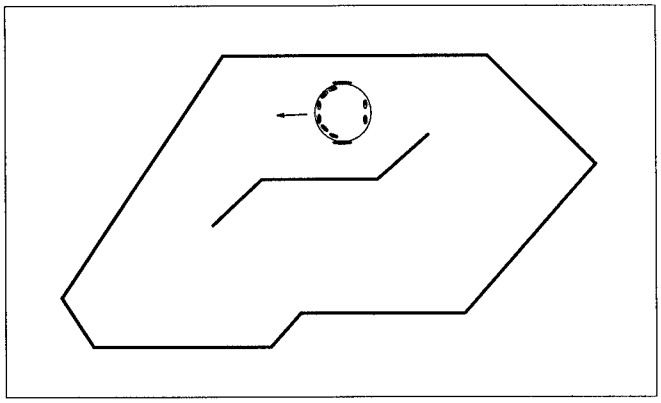


Figure 4: The arena of approximate size 60x45 cm. The indicated position (facing left) is the starting point for fitness evaluation.

The fitness is measured as follows. At a particular time step t, both motor speeds V_l and V_r as well as all eight proximity sensors IR_i are measured. Then, the speed of the robot's center $V_t = (V_l + V_r)/2$, the penalty term $\Delta v_t = |V_l - V_r|$, and the sensor with the highest activation $\hat{IP} = \max_i IP_i$ is calculated. The fitness contribution f_t for step t is then

$$f_t = V_t(1 - \sqrt{\Delta v_t})(1 - \hat{IP_t}) \quad . \tag{3}$$

Finally, the total fitness is the sum over $t_{\max}$ (e.g., 240) time steps

$$F = \sum_{t=1}^{t_{\max}} f_t = \sum_{t=1}^{t_{\max}} V_t(1 - \sqrt{\Delta v_t})(1 - \hat{IP_t}) \quad . \tag{4}$$

Khepera's on-board controller allows for sending all sensor values every each 100 milliseconds (ms). That means, the robot moves for 100 ms with constant motor speeds. Then, the controller receives the new values and calculates new motor speeds for the next time step. Meanwhile, the individual fitness component f_t is calculated and added to the total fitness.

4 Experiments

This section reports some typical results when evolving Braitenberg vehicles by means of a $(3, 6)$-ES with self-adaptation of the step size [24]. Some of the results are discussed in Subsection 4.3. A $(3, 6)$-ES generates 6 new offspring per generation and selects the 3 fittest individuals as parents for the next generation. A $(3, 6)$-selection scheme implies that the strategy does not use any elitist selection scheme. A non-elitist selection scheme was chosen, since the fitness evaluation is extremely noise (cf. Fig. 2). In the experiments reported in this paper, the ES generates two offspring without crossover, two with uniform recombination, and two with intermediate recombination, and the initial standard deviation was set to

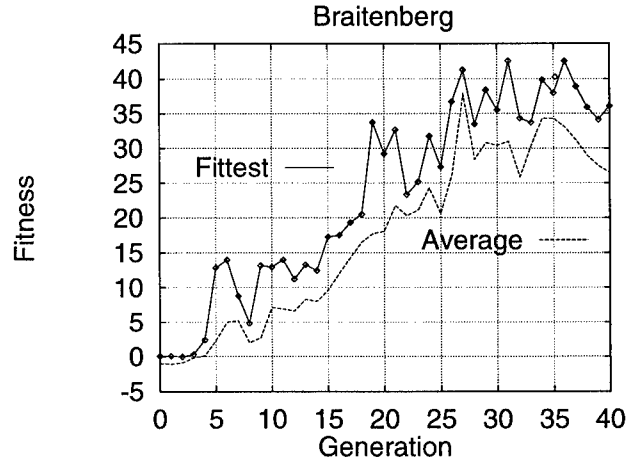


Figure 5: A typical run of the evolution of a *constrained* Braitenberg vehicle. The upper graph represents the best individual whereas the lower graph prepresents the average of the whole population.

0.5. A further source of additional noise is that also the individual's starting conditions vary. Over a long time, the environmental conditions cannot be hold constant; this phenomenon is intrinsic to real-world applications.

For each experiment, at least four runs have been performed. Figures 5 and 6 present typical runs that resemble average performance. Each figure shows the fitness of the best individual as well as the average fitness of the entire population. No average of several runs is shown, since averaging eliminates interesting details. The other runs are within a 20 percent interval of the presented runs.

In the evolution of Braitenberg vehicles, a main problem is to find a first controller that exhibits an even tiny reasonable behavior. Initializing all weights at random leads to an agent that immediately crashes into the wall, where it gets stuck. To help guide the evolution process, all weights were initialized with very small negative random weights w_i^l and w_i^r (i.e., $[-0.5, 0]$) and the weights w_0^l and w_0^r were set to very small positive values (i.e., $[0, 0.1]$). Furthermore, the first series of experiments exploit the morphology of the robots, i.e., the left and right part of the controllers are constrained to be equal $w_0^l = w_0^r$ and $w_i^l = w_{(14-i) \bmod 8+1}^r$. This constraint leads to a reduced search space of nine parameters. In all experiments, fitness evaluation was done over 240 time steps. Since one time step requires 100 ms, the evaluation of each controller takes about 24 seconds.

4.1 Constrained Controllers

A typical run of the evolution of a *constrained* Braitenberg controller can be seen in Fig. 5. Figure 5 shows the fitness of the population's fittest agent and the average of the whole population. In the first few genera-

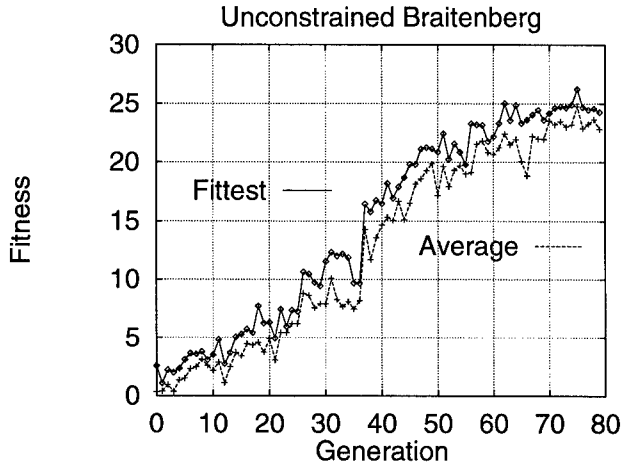


Figure 6: A typical run of the evolution of a *unconstrained* Braitenberg vehicle. The upper graph represents the best individual whereas the lower graph prepresents the average of the whole population.

tions, most vehicles are sitting at the initial position or crashing backwards into the wall resulting in a negative fitness. After a few more generations, some agents are moving forward, but after some steps, got stuck at the wall; at that time, the avoidance behavior was not sufficient. However, such a short forward movement results in a small positive fitness, which is a first step towards a useful agent. After about eight to ten generations, the fittest agents are moving slowly inside the corridor. But sometimes, they are hitting the walls again. After hitting a wall, good controllers set the motor speeds to negative values, which cause the robots to back up from the wall, and after turning, they continue moving inside the corridor. After 30 generations, the fittest agents perform up to three complete laps in the arena. Even though both controller sides are constrained to be equal, identical weights do not ensure total symmetric behavior. Tolerances in the electrical characteristics of the motors and sensors impose an unsymmetric behavior. Consequently, such controllers have to find a good compromise. As a result, Braitenberg vehicles with constrained controllers move forward in an almost straight line and turn, if they approach an obstacle. Overall, a Braitenberg vehicle with a constrained controller moves with a high speed but with a rather rough trajectory. But a Braitenberg vehicle adapts its behavior to both the environment and the unspecified or even changing characteristics of its electrical components.

4.2 Unconstrained Controllers

Figure 6 shows a typical run of the evolution of an *unconstrained* Braitenberg controller. Such a controller has 18 parameters and the ES has to evolve both sides of the controller, i.e., the connections for the left motor activation M^l as well as the right motor activation M^r. Thus, as can be seen in Figs. 5 and 6, the evolution of an unconstrained Braitenberg controller requires more time, and also, the final performance is lower than the performance of the constrained controller. The unconstrained controller develops a different survival strategy. From the very beginning, the first controllers have different bias weights w_0^l and w_0^r, which causes the robot to turn in small circles. Circling around results in a small positive fitness. During the next generations, this circling process is preserved, but the radius becomes larger and larger. After approximately 30 generations, one controller side improves its object-avoidance behavior so that it prevents the robot from crashing into the wall; the resulting behavior can be interpreted as wall following. This wall-following behavior is not encoded in the fitness function. Rather, it is emerged from the evolution process. In the ongoing evolution process, this wall-following behavior is further improved, and after about 80 generations, most robots perform two complete laps in the arena.

4.3 Discussion

The previous two subsections have investigated the development of two different controller types, namely constrained and unconstrained controllers. The constrained control architecture is inspired by the biological observation that most animals are of symmetric shape. A constrained controller has less parameters, which accelerates the evolution process, but, at the same time, limits the number of potential solutions. As a consequence, both controllers develop different survival strategies.

The relations between both controller types were further investigated in a series of control experiments, in which the weights of unconstrained controllers were initialized with weights taken from an evolved constrained controller. In the first five to ten generations, these controllers were loosing their behavior and the fitness dropped to values around 10. Afterwards, these unconstrained controllers start develop the wall-following behavior as already discussed. Thus, the developed controller is adapted not only to the environment but also to the control architecture itself.

Further investigations have shown that a controller reacts very sensitive to parameter changes; the controller's parameters exhibit high epistatis. This epistatis results from the sensor's non-linearities and the 100 ms time interval between two subsequent sensor readings. It is not reasonable to change, for example, only the bias weight w_0^l. This would result in a too high speed and the robot would crash into the next wall. Conversely, changing only one "avoidance" weight w_i^l would degrade the fitness, since the robot would tend to wriggle. Consequently, all parameters have to be adapted simultaneously. The ES obeys this requirement, by using a mutation probability $p_m = 1.0$, i.e., the ES applies muta-

tion to all parameters at the same time. It is suspected that the observable epistatis is the main reason for the inefficiency of GA-based approaches [8, 9, 18]. That research reports that the GA needs about 50 to 100 generations with a population of 80 individuals. Such an optimization process takes approximately 66 hours, which is approximately 40 times longer than that of the ES approach. This coincides with the research discussed in [22], which investigates the performance of GAs when applied to artificial fitness functions. The main results presented in [22] indicate that the independence of the parameters to be optimized is an essential prerequisite for GAs and that the GA's performance significantly degrades under epistatis.

Even though this paper points to other GA-based research, the main focus of this paper is the application of the ES to the evolution and optimization of Braitenberg vehicles. A comparison acros remote research would be very problematic, since not all parameters of such a real-world application can be replicated. In addition, we did a series of control experiments, in which we used a GA instead of the ES. The control experiments yield roughly the same performance as reported in [8, 9, 18]. The GA needs approximately ten times more fitness evaluations than the ES. The GA suffers from the high epistatis and the need of a rather large population of about 80 individuals. In each generation, the GA generates ten times more offspring than the ES. Thus, in this application, the ES converges in a time period, in which the GA performs only five to ten generations.

5 Enhancements

For the research on GAs it is very important to consider genomes with varying length. Several applications, e.g., [14], explicitly use this feature for the development of more complex control structures. The main underlying idea is that first, the GA develops a small solution with the most important properties. In the ongoing evolution process, the genome is allowed to grow in size, which enables the system to add more beneficial features.

It is often argued that this dynamical growth of the genome is proprietary for GAs and not reasonable for the ES. However, as outlined above, both types of algorithms are very similar. Both apply mutation as well as recombination. For a mutation operator, the length of the genome does not matter. If applying recombination/crossover, these operators have to ensure the correctness of a new genome anyway.

In [21] a hybrid method has been proposed that allows for the development of neural networks with minimal topology. Essentially, this hybrid method works as follows. Each offspring randomly adds and removes neurons as well as connection from the network that is has inherited from its parents. Similar to the self-adaptation of the step size, this hybrid method self-adapts the adding and removing probabilities. By these means, the network can dynamically grow and shrink depending on the actual environment.

6 Conclusions

This paper has discussed the practical application of the evolution strategy to the evolution and optimization of Braitenberg vehicles. Braitenberg vehicles are autonomous agents with a simple control architecture, which is typically implemented as a neural network. Autonomous agents are very important tools in New Artificial Intelligence, since they study intelligence as the result of a system environment interaction, rather than understanding intelligence on a computational level.

In the practical experiments, constrained as well as unconstrained Braitenberg controllers have been investigated. These two controllers typically develop different survival strategies, which have also been discussed. A comparison with other research that apply genetic algorithms to very similar tasks shows that the ES-based approach is much faster than the GA-based approach. It is suspected that the high epistatis between the controller's parameters drastically slows down the GA-based approach. The ES speeds up the development of Braitenberg controllers by more than one order of magnitude, which is very important, since experimentation is to be done with real system. The ES converges in a few hours compared to 66 hours required for the GA-based approach.

This paper has also argued that the evolution strategy is more adaptive, since it self-adapts parameters, such as the step size or mutation probability. This allows to better embed the whole approach in more complex tasks. It could be argued that the small mutation rate as is usually used in GAs is biologically more plausible than a mutation rate $p_m = 1$ as is used in the ES. However, we argue that it is the other way around. If one looks at living thinks, all individuals differ in almost all attributes. For example, if comparing two arbitrary selected humans, these two humans will differ in all perspectives. It seems that nature applies mutation to only a few genes. However, most genomes do not encode all parameters of the resulting individual. Rather, the genome encodes developmental processes. Thus, modifying one gene results in different developmental programs, and consequently, the resulting individual differs in (almost) all perspectives. Since, most evolutionary approaches do not involve real developmental processes, such as growing, we argue that the ES better reflect nature's principles.

Furthermore, the experiments also indicate that the evolution process can highly benefit from an exploitation of physical matters.

Further research will be devoted to more complex control architectures for object avoidance, navigation, and manipulation as well as other neural controllers,

which are designated to enhance the robots capabilities/competences

Acknowledgements

This work was supported in part by a Human Capital and Mobility fellowship of the European Union, grant number ERBCHBICT941266. Thanks to Rolf Pfeifer and Peter Eggenberger for helpful discussion.

References

[1] Bäck, T. Optimal Mutation Rates in Genetic Search. In S. Forrest (ed.) *Proceedings of the Fifth International Conference on Genetic Algorithms*, pp. 2-8. Morgan Kaufmann, San Mateo, CA, 1993.

[2] Bäck, T. and Schwefel, H.-P. An Overview of Evolutionary Algorithms for Parameter Optimization, *Evolutionary Computation* 1(1), pp. 1-23. MIT Press, Cambridge, MA, 1993.

[3] Beyer, H.-G. Toward a Theory of Evolution Strategies: On the Benefits of Sex – the $(\mu/\mu, \lambda)$ Theory. *Evolutionary Computation* 3(1), pp. 81-111. The MIT Press, Cambridge, MA, 1995.

[4] Braitenberg, V. *VEHICLES, Experiments in synthetic psychology*. MIT-Press, Cambridge, Massachusetts, 1984.

[5] Brooks, R. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, **2**, pp. 14-23, 1986.

[6] Cliff, D., Husbands, P., Harvey, I. Evolving Visually Guided Robots. In J.-A. Meyer, H.L. Roitblat, and S.W. Wilson (eds.), *From animals to animats 2*, pp. 374-383. MIT Press, Bradford Books, Cambridge, MA, 1992.

[7] De Jong, K.A. *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*. Ph.D. Thesis, University of Michigan, 1975.

[8] Floreano, D. and Mondada, F. Automatic Creation of an Autonomous Agent: Genetic Evolution of a Neural-Network Driven Robot. In D. Cliff, P. Husbands, J. Meyer, and S.W. Wilson (eds.), *From Animals to Animats III: Proceedings of the Third International Conference on Simulation of Adaptive Behavior*, pp. 421–430. MIT Press, Bradford Books, Cambridge, MA, 1994.

[9] Floreano, D. and Mondada, F. Evolution of Homing Navigation in a Real Mobile Robot, to appear in: *IEEE Transactions on Systems, Man, and Cybernetics-Part B*, Vol.26, No.3, June 1996.

[10] Fogel, L.J. "Autonomous Automata", *Industrial Research*, vol. 4, pp. 14-19, 1962.

[11] Fogel, D.B. *Evolutionary Computation: Toward a New Philosophy of Machine Learning Intelligence*, IEEE Press, Piscataway, NJ, 1995.

[12] Goldberg, D.E. *Genetic Algorithms in Search, Optimization and Machine Learning* Addison-Wesley Publishing Company, 1989.

[13] Holland, J.H. *Adaptation in Natural and Artificial Systems* Ann Arbor, Michigan, University of Michigan Press, 1975.

[14] Harvey, I., Husbands, P., Cliff, D., Thompson, A., and Jakobi, N. Evolutionary Robotics: the Sussex Approach. In R. Pfeifer and R. Brooks (eds.), *Robotics and Autonomous Systems, Special Issues on "Practice and Future of Autonomous Agents"*, 1996.

[15] *Khepera Users Manual*, Laboratoire de microinformatique, Swiss Federal Institute of Technology (EPFL), 1015 Lausanne, Switzerland.

[16] P. Maes (ed), *Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back*. The MIT Press, Cambridge, MA, 1991.

[17] Mühlenbein, H. and Schlierkamp-Voosen, D. The Science of Breeding and its Application to the Breeder Genetic Algorithm. *Evolutionary Computation* 1(4), pp. 335-360. The MIT Press, Cambridge, MA, 1993.

[18] Nolfi, S. and Parisi, D. Learning to Adapt to Changing Environments in Evolving Neural Networks, Technical Report 95-15, Institute of Psychology. National Research Council, Rome, Italy. WWW http://kant.irmkant.rm.cnr.it/public.html. 1995.

[19] Potter, M.A. and De Jong, K.A. A Cooperative Coevolutionary Approach to Function Optimization. In: Y. Davidor, H.P. Schwefel, and R. Männer (eds.), *Proceedings of Parallel Problem Solving from Nature 3*, pp. 249-257. Springer-Verlag, 1994.

[20] Rechenberg, I. *Evolutionsstrategie*. Frommann-Holzboog, Stuttgart, 1973.

[21] Salomon, R. A Hybrid Method for Evolving Neural Network Topologies. In C.H. Dagli, B.R. Fernández, J. Ghosh, and R.T.S. Kumara (eds.), *Proceedings of the Artificial Neural Networks in Engineering (ANNIE'94)*, pp. 147-152. New York: ASME Press, 1994.

[22] Salomon, R. Performance Degradation of Genetic Algorithms under Coordinate Rotation. To appear in *Fifth Annual Conference on Evolutionary Programming EP'96*, to be held at San Diego, USA, February 29 - March 3, 1996.

[23] Salomon, R. Implicit Independence Assumptions; a Notorious Problem for Genetic Algorithms. To appear in *Proceedings of the International Symposia on Soft Computing and Intelligent Industrial Automation SOCO'96*, to be held in Reading, UK, March 26 - March 28, 1996.

[24] Schwefel, H.P. *Evolution and Optimum Seeking*. John Wiley and Sons, Inc, New York, Chicester, Brisbane, Toronto, Singapore, 1995.

[25] L. Steels (ed.), The Biology and Technology of Intelligent Autonomous Agents. Special issue of *Robotics and Autonomous Systems* **15** Elsevier, Amsterdam, Lausanne, New York, Oxford, 1995.

[26] Thierens, D. and Goldberg, D. Convergence Models of Genetic Algorithm Selection Schemes. In Y. Davidor, H.P. Schwefel, and R. Männer (eds.), *Proceedings of Parallel Problem Solving from Nature 3*, pp. 119-129. Springer-Verlag, 1994.

Toward the Evolution of Dynamical Neural Networks for Minimally Cognitive Behavior

Randall D. Beer

Santa Fe Institute
1399 Hyde Park Rd.
Santa Fe, NM 87501
beer@santafe.edu

Dept. of Computer Engineering and Science
and Dept. of Biology
Case Western Reserve University
Cleveland, OH 44106
beer@alpha.ces.cwru.edu

Abstract

Current debates regarding the possible cognitive implications of ideas from adaptive behavior research and dynamical systems theory would benefit greatly from a careful study of simple model agents that exhibit minimally cognitive behavior. This paper sketches one such agent, and presents the results of preliminary experiments on the evolution of dynamical neural networks for visually-guided orientation, object discrimination and accurate pointing with a simple manipulator to objects appearing in its field of view.

1 Introduction

Many of the key ideas emphasized in adaptive behavior research are beginning to have a significant impact on cognitive science. For example, adaptive behavior research in general, and the dynamical perspective on adaptive behavior that is often taken in such research in particular, have begun to significantly influence the growing debates concerning the nature and necessity of notions of representation and computation in explaining cognitive behavior (Brooks, 1991; Clark & Toribio, 1994; Beer, 1995b; Port & van Gelder, 1995). Likewise, the important roles played by an agent's body and its environment in the generation of its behavior, long emphasized in adaptive behavior research, parallel a renewed concern for embodiment and situatedness in cognitive science (Suchman, 1987; Lakoff, 1987; Damasio, 1994; Haugeland, 1995; Hutchins, 1995). The sorts of decentralized and distributed control mechanisms that are utilized in adaptive behavior research are also strongly reminiscent of the picture of human brain activity that is emerging from neuroanatomical and brain imaging studies in cognitive neuroscience (Posner & Raichle, 1994; Gazzaniga, 1995), as well as from more detailed studies of invertebrate nervous systems (Altman & Kien, 1989). Finally, dynamical and adaptive behavior ideas are beginning to significantly impact work in developmental psychology (Rutkowska, 1994; Thelan & Smith, 1994).

Despite this widespread impact on cognitive science, most of the empirical work in adaptive behavior research to date has focused on relatively simple sensorimotor behavior, such as obstacle avoidance and wall following. While there are clear advantages to this strategy, and much is still not understood about the design and analysis of even these simple behaviors, it is unfortunate that so much of the discussion concerning the cognitive implications of adaptive behavior ideas is being carried out in the absence of concrete models. For example, current debate on the role of representation in cognition is mostly occurring at a philosophical level, with intuition and analogy rather than the careful study of concrete models leading the way. But one of the major advantages of an adaptive behavior approach, particularly an evolutionary one, is that, by grounding an agent's behavior in an environment and making far fewer a priori assumptions about the necessary design of its internal control mechanisms, this approach provides a much broader intellectual playing field on which to explore these issues through the development and analysis of concrete models.

My own recent work in this area has focused on the use of evolutionary algorithms to evolve dynamical neural networks for controlling the behavior of model agents and then analyzing the dynamics of the resulting agent-environment systems. Here, an agent's nervous system is viewed not as an information processing system, but rather as a dynamical system which, in conjunction with the dynamics of the agent's body and its environment, is capable of producing effective behavior in that environment. To date, this approach has been successfully applied to chemotaxis, legged locomotion, sequential behavior and learning (Beer & Gallagher, 1992; Yamauchi & Beer, 1994). The principal motivation behind this work has been the development and analysis of simpler idealized models of adaptive behavior for the purpose of elucidating the essential principles of a dynamical theory of adaptive behavior (Beer, in press). To this end, dynamical analyses have been performed on many of these evolved circuits, and a preliminary theoretical framework for adaptive behavior has been proposed (Beer, 1995a; Beer, 1995b).

The goal of the work described in this paper is to begin to explore the applicability of this approach to the design and analysis of more cognitive behavior. Section 2 sketches a visually-guided agent whose capabilities are both rich enough to begin to explore cognitive behavior yet simple enough to be tractable to evolution and analysis. The particular agent, neural network model and evolutionary

algorithm used here are described in Section 3. Sections 4-6 describe the results of preliminary experiments in the evolution of dynamical neural networks for visually-guided orientation, object discrimination and accurate pointing, respectively. Finally, related work is briefly discussed in Section 7 and Section 8 summarizes the results of this paper and suggests some directions for future research.

2 A Visually-Guided Agent

What sorts of agents and behaviors should we attempt to study? On the one hand, the capabilities and behavior of the model agents that we study must be rich and sophisticated enough to be cognitively interesting, so that they raise the sorts of issues that we would like to explore. For example, if we wish to explore the nature and necessity of the notion of representation in cognitive behavior, then we must examine tasks that are sufficiently "representation-hungry" (Clark & Toribio, 1994). On the other hand, these model agents must be simple enough to be computationally and analytically tractable, so that we have some hope of evolving and analyzing them using techniques that are at most an incremental step beyond what is currently known to be feasible. The term "minimally cognitive behavior" is meant to connote the simplest behavior that raises cognitively interesting issues.

Generally speaking, visually-guided behavior provides an excellent arena in which to explore the cognitive implications of dynamical and adaptive behavior ideas, since it raises a host of issues of immediate cognitive interest. Visually-guided behavior includes such phenomena as visual orientation, object perception and discrimination, visual attention, perception of self-motion, object-oriented action, and visually-guided motion and manipulation. However, despite this complexity and richness, significant progress on understanding the processes and neural architectures underlying visually-guided behavior is beginning to be made in cognitive neuroscience (Posner & Raichle, 1994; Gazzaniga, 1995). Furthermore, a relatively simple model agent can be designed that supports simplified versions of all of these phenomena.

The model agent is illustrated in Figure 1. This two-dimensional agent possesses an "eye" consisting of a foveated array of distance sensors, two "motors" that produce 2D movement of the entire body, and a simple transparent 2 degree-of-freedom "arm" (rotation about the body and extension along its length) and opaque 1 degree-of-freedom "hand" (rotation about the "wrist") for manipulating objects. Note that the intent here is not to model in any depth the particular visually-guided behavior of any real animal or robot. Rather, the goal is to explore the space of possible dynamical organizations of agents that engage in minimally cognitive behavior. Thus, there is no particular need to strive for physical realism in these experiments. For example, the agent's "vision" is certainly not intended as a serious model of the actual physics of light or photoreception. However, it does raise some of the same issues in the perception of objects using a spatially-structured array of distal sensors. Likewise, while the agent's hand does not realistically model the limbs of any animal or robot, it does raise analogous issues in the visual control of manipulation.

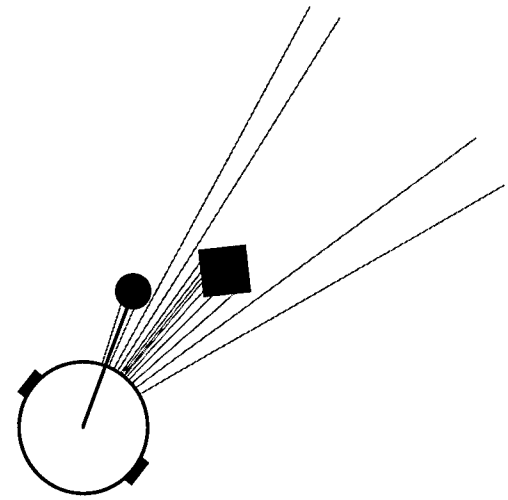


Figure 1: Basic design of a visually-guided agent. The agent (large circle) has an eye (gray lines), two motors (filled rectangles) and a transparent arm (solid line) with an opaque hand (filled circle). The arm can rotate about the center of the agent and extend or retract along its length. The hand can rotate about its point of attachment with the arm.

Given its sensory and motor capabilities, what sorts of cognitively interesting behavior might this agent engage in? This agent could perceive the two-dimensional structure of objects and organize its behavior in accordance with their shapes (e.g., orienting to a novel object and discriminating one object from another). It could navigate around obstacles in two dimensions, deciding which gaps its body can fit through and which it cannot. It could also exhibit simple forms of object persistence (e.g., continuing to pursue a goal object that is momentarily occluded by another object). At a more sophisticated level, selectively interacting with one object from among a set of objects raises interesting focus-of-attention issues. In addition, the fact that its hand is opaque raises interesting issues in the discrimination of self from nonself. This agent could also actively manipulate objects in its environment (e.g., building simple structures out of the objects in its environment). Finally, one could imagine groups of these agents engaging in simple cooperative tasks, such as tossing an object back and forth. Thus, the behavior of this agent is potentially of some cognitive interest.

The remainder of this paper describes three sets of experiments aimed at an initial exploration of some of the simplest capabilities of the agent sketched above. Specifically, these experiments are designed to establish the basic

soundness and feasibility of the proposed agent, to explore how difficult it is to evolve dynamical neural network controllers for these tasks, and to determine the neural architectures and evolutionary algorithm configurations that are best suited to them.

3 Methods

In all of the following experiments, the agent has a circular body with a diameter of 30 (in an environment of size 400 × 275), with an eye consisting of either 5 or 7 rays of maximum length 220 uniformly distributed over a visual angle of $\pi/6$.[1] An intersection between a ray and an object causes an input to be injected into the corresponding sensory neuron. The magnitude of the injected input is inversely proportional to the distance to the object. When rays are at their maximum length, no input is injected, while a maximum input of 10 is injected for rays of zero length.

The spatial resolution of the agent's eye is determined by a number of factors. Resolution obviously depends on the number of rays and the visual angle over which they are distributed. Resolution also clearly depends on how far away an object is, since a more distant object will intersect fewer rays. Finally, the spatial resolution of the eye is very dependent on the values of the bias and gain parameters of the ray sensory neurons. If the biases are too high or too low, then objects will give either a saturated response or no response, respectively. If the gains are too low, each ray will show very little difference in response regardless of its length. If the gain is too high, each ray will essentially give a binary response at a very narrow range of distances (which will make the ray biases very difficult to evolve). While these issues will be discussed in greater detail in the pointing experiments described in Section 6, they are important to keep in mind for all of the experiments described in this paper.

The agent's behavior is controlled by a continuous-time recurrent neural network (Beer, 1995c) with the following state equation:

$$\tau_i \dot{y}_i = -y_i + \sum_{j=1}^{N} w_{ji}\sigma\left(g_j(y_j + \theta_j)\right) + I_i \quad i = 1,\ldots,N$$

where y is the state of each neuron, τ is its time constant, w_{ji} is the strength of the connection from the j^{th} to the i^{th} neuron, g is a gain, θ is a bias term, $\sigma(x) = 1/(1+e^{-x})$ is the standard logistic activation function, and I represents an external input (e.g., from a sensor). States were initialized to 0 and circuits were integrated using the forward Euler method with an integration step size of 0.1.

[1] As in any simulation which is not intended as a literal model of the real world, the actual units are essentially arbitrary. For concreteness, one can assume that distances are in cm, time is in seconds, and velocities are in cm/sec.

The evolutionary algorithm used in the experiments described in this paper is similar to a very simple evolutionary strategy (Bäck & Schwefel, 1993). A population of individuals is maintained, with each individual encoded as a vector of real numbers (representing the connection weights w_{ji}, the biases θ_i and the time constants τ_i, as well as the gains g_i of the ray sensory neurons, with all other gains fixed to 1). Initially, a random population of vectors is generated by initializing each component of every individual to random values uniformly distributed over the range ±1. Individuals are selected for reproduction using fitness proportional selection with linear fitness scaling with a fitness scaling multiple of 2 (Goldberg, 1989). A selected parent is mutated by adding to it a random displacement vector whose direction is uniformly distributed on the M-dimensional hypersphere (Knuth, 1981, p. 130) and whose magnitude is a Gaussian random variable with 0 mean and variance σ^2. For each slot in the new population, the child is chosen if its performance is greater than or equal to that of the parent, otherwise the parent is copied. All random numbers were generated using the routine `ran1` described in (Press et al, 1994, p. 280), which has a period greater than 10^8 and uses a shuffling algorithm to remove low-order serial correlations.

In the experiments described in this paper, search vector components were mapped to circuit parameters using linear maps from ±1 to a given range of circuit parameter values. Unless otherwise stated, these circuit parameter ranges were as follows: circuit biases ∈ [-5,5], time constants ∈ [1,2], and connection weights ∈ [-5,5]. Because ray sensor gain and bias ranges varied across experiments, they will be reported separately below. Gains were clipped to be greater than 0 and time constants were clipped to be greater than 1.

4 Orientation Experiments

One of the most basic capabilities required by any visually-guided agent is the ability to orient to a visual stimulus. In the first set of experiments to be described, agents were evolved that could use their vision to adjust their horizontal position so as to catch falling objects (Figure 2).

These agents had 5 rays. Their horizontal velocity was proportional to the sum of the opposing horizontal forces produced by each motor (with a constant of proportionality of 0.2). Circular objects with a diameter of 26 were dropped from the top of the environment with an initial horizontal offset from the center of the agent in the range ±70, a horizontal velocity in the range ±6, and a vertical velocity in the range [0.5,5].

The performance measure to be maximized was:

$$200 - \sum_{i=1}^{NumTrials} d_i \Big/ NumTrials$$

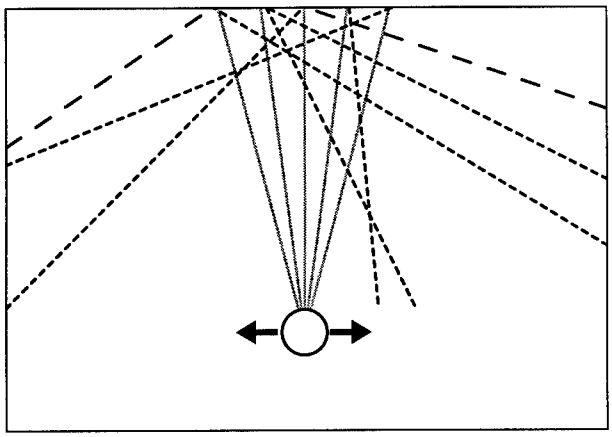


Figure 2: Experimental setup for orientation experiments. The agent moves horizontally. Its rays are shown in gray. Dotted and dashed lines denote the paths of circular objects used to evaluate the agent's performance during evolution, as described in the text.

where *NumTrials* is the total number of trials and d_i is the horizontal distance between the centers of the object and the agent when their vertical separation goes to 0 on the i^{th} trial.

In the first set of orientation experiments, bilaterally symmetric feedforward networks with 5 ray sensory neurons and 2 motor neurons were evolved (for a total of 8 parameters). All time constants were fixed to 1. Ray sensor biases were in the range [-10,-5] and ray sensor gains were in the range [1,5]. All ray sensory neurons shared the same gain and bias. Populations of 25 individuals were evolved for 50 generations with a mutation variance σ^2 of 10. Six evaluation trials were used (shown as dotted lines in Figure 2). Note that when objects reach one of the walls, their horizontal motion stops but their vertical motion continues.

This turned out to be a fairly simple task, and agents with a mean fitness of 99.31% (N = 5) quickly evolved.[2] The movement of a typical agent on one trial is shown in Figure 3a. Note how the agent quickly orients to the object and then tracks it as it falls. Interestingly, these agents generalized poorly to 100 random trials, with mean fitness dropping to 79.60%. An examination of the qualitative behavior of these agents revealed that they were failing to respond quickly enough to objects with large horizontal and small vertical velocities.

In an attempt to improve this deficiency, two additional evaluation trials were used (shown as dashed lines in Figure 2) and the 5 experiments were repeated. The resulting agents had a mean fitness of 87.29% on the 8 evaluation trials, and a mean fitness of 90.25% on 100 random trials. Closer examination revealed that these new agents were indeed more sensitive to objects with large horizontal and small vertical velocities, primarily because the mean bias of their ray sensors (-0.85 ± 0.18 s.e.) was significantly larger than the mean bias of the ray sensors in the first set of agents (-2.06 ± 0.33 s.e.) ($p < 0.02$ using Welch's approximate t-test). However, these agents still missed some of these objects because they would quickly pass out of the agent's field of view and these agents would not pursue objects that they could no longer see. An example of this problem is shown in Figure 3b.

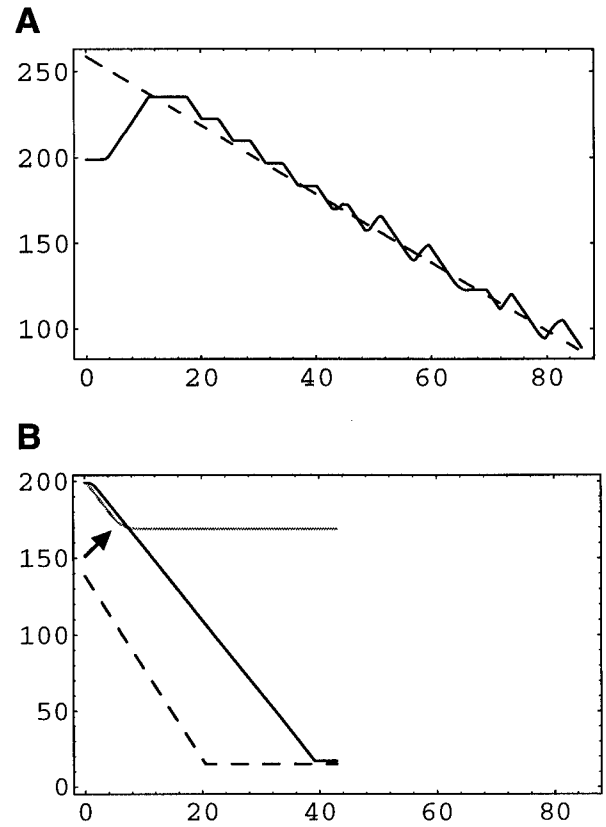


Figure 3: Plots of the horizontal positions versus time of evolved orientation agents (solid lines) attempting to catch a circular object (dashed lines). (a) Path of a typical reactive agent on a successful trial. (b) A comparison of the paths taken by typical reactive (gray line) and dynamical (black line) agents for an object with large horizontal and small vertical velocity. While the reactive agent ceases to move as soon as the object leaves its visual field (arrow), the dynamical agent continues to pursue the object, eventually catching it against the left wall.

Thus, these agents are faced with a simple example of an object persistence problem. Since both the agent and the objects are constrained to remain within the "walls" of this environment, these agents should continue to pursue objects that have momentarily passed out of their field of view. However, because these agents are controlled by feedforward networks, they are purely reactive; they cannot organize their behavior according to sensory stimuli that are no longer present. Note that adding any number of interneurons is not going to solve this problem. Rather, what is needed is for the circuits controlling these agents to have internal dynamics.

[2] Throughout the paper, fitness will be reported as a percentage of the maximum attainable performance.

In order to address this object persistence problem, a final set of orientation experiments was run using a dynamical elaboration of the feedforward circuit used earlier, now with evolvable time constants and bilaterally symmetric self and recurrent connections between the motor neurons (for a total of 12 parameters). All other aspects of these experiments were identical to the second set of experiments. In this case, the resulting agents had a mean fitness of 99.09% on the 8 evaluation trials and a mean fitness of 96.60% on 100 random trials. An examination of the qualitative behavior of these dynamical agents revealed that they would indeed continue to pursue objects that had momentarily disappeared from their field of view (Figure 3b). Similar results were obtained with circuits in which the ray sensory neurons and the motor neurons were fully interconnected. Thus, even in this relatively simple task and circuit, internal dynamics can offer significant advantages to an agent by allowing its behavior to depend not only on its immediate circumstances, but also on its recent history of interaction with the environment.

5 Discrimination Experiments

In order to selectively interact with different objects, a visually-guided agent must be capable of visually discriminating between them. In a second set of experiments, agents were evolved which could discriminate between circles and diamonds and between circles and horizontal lines, catching circles as in the orientation experiments while avoiding the other objects (Figure 4).

These agents had 7 rays. The experimental setup was similar to that used in the orientation experiments, with agents moving horizontally as objects fall from above. In this case, objects fell straight down with an initial horizontal offset in the range ±50 and a vertical velocity of either 3 or 4. Circular objects had a diameter of 30, diamonds had sides of length 30 and lines had a length of 30.

The performance measure to be maximized was:

$$\sum_{i=1}^{NumTrials} p_i \Big/ NumTrials$$

where $p_i = 1 - d_i$ for a circular object and $p_i = d_i$ for the other objects, d_i is the horizontal distance between the centers of the object and the agent when their vertical separation goes to zero on the i^{th} trial (clipped to *MaxDistance* and normalized to run between 0 and 1), *NumTrials* is the total number of trials, and *MaxDistance* is 1.5 times the sum of the radii of the object and the agent. The reason that d_i was clipped to *MaxDistance* was to prevent the avoidance of, for example, diamonds by large distances from dominating the fitness at the expense of accuracy in catching circles. A total of 24 evaluation trials were used during evolution, uniformly distributed over the range of horizontal offsets and alternating between circular objects and either diamonds or lines (Figure 4). The large number of trials was necessary in order to ensure good generalization.

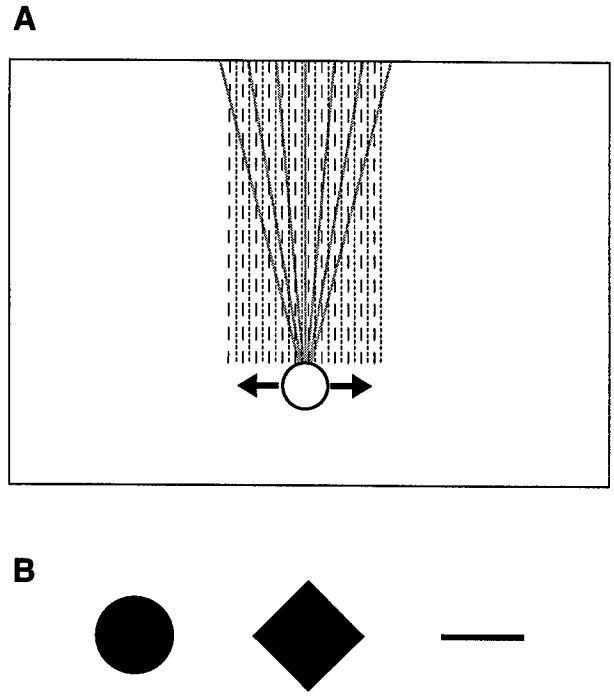


Figure 4: Experimental setup for discrimination experiments. (a) The agent moves horizontally. Its rays are shown in gray. Broken lines denote the paths of circular (dotted) and other (dashed) objects used to evaluate the agent's performance during evolution. (b) Evaluation objects used in the discrimination experiments.

The circuit architecture was bilaterally symmetric, with 7 ray sensory neurons projecting to 5 fully interconnected interneurons which in turn projected to the two motor neurons controlling horizontal motion (for a total of 47 parameters). Ray sensor gains were in the range [1,5]. Many different ray sensor bias ranges were used, from [-10,0] to [-4,-2], with the narrower ranges generally giving better results. All ray sensory neurons shared the same gain and bias. All other ranges were identical to those used in the orientation experiments.

Agents that could visually discriminate between objects were much more difficult to evolve than agents that could simply orient to an object. In these experiments, populations of from 300 to 400 individuals were evolved for from 100 to 200 generations with a mutation variance σ^2 of 15. The behavior of the best discriminator of circles and diamonds and the best discriminator of circles and lines will be described here.

The best circle/diamond discriminator had a mean fitness of 99.83% on the 24 evaluation trials and a mean fitness of 98.96% on 100 random trials. Qualitatively, all objects were correctly classified on the 24 evaluation trials and only

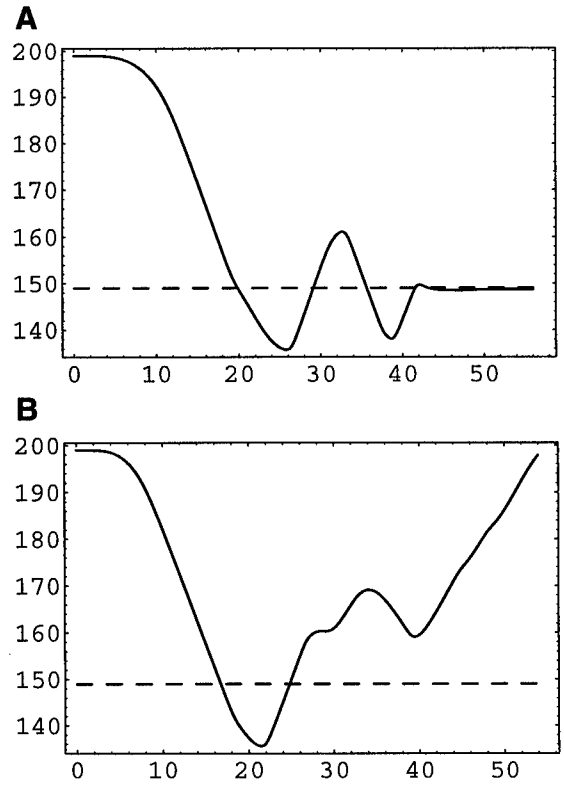


Figure 5: Plots of the horizontal positions over time of an evolved agent that can visually discriminate between circles and diamonds. The path of the agent is shown as a solid line, while that of the object is shown as a dashed line. (a) Path of the agent catching a circle. (b) Path of the agent avoiding a diamond.

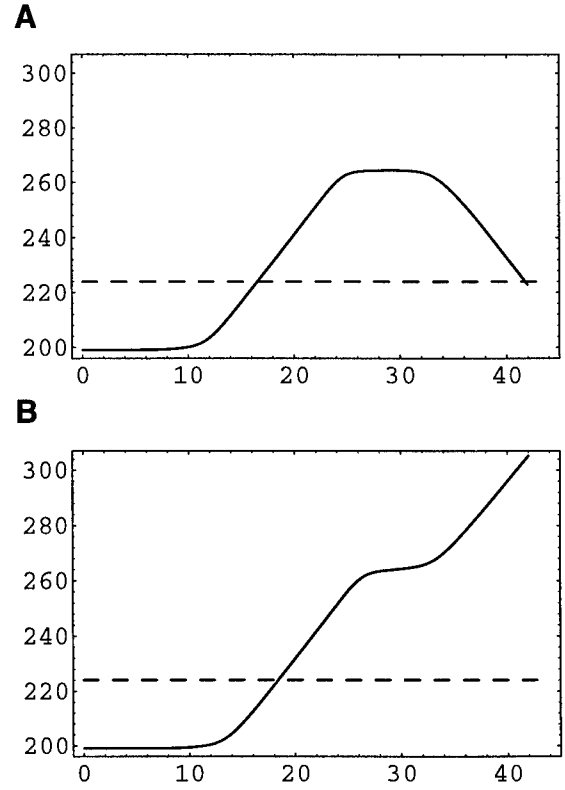


Figure 6: Plots of the horizontal positions over time of an evolved agent that can visually discriminate between circles and lines. The path of the agent is shown as a solid line, while that of the object is shown as a dashed line. (a) Path of the agent catching a circle. (b) Path of the agent avoiding a line.

one circle was incorrectly classified during the 100 random trials.

The behavior of this agent is shown in Figure 5, both while catching a circle (Figure 5a) and while avoiding a diamond (Figure 5b) dropped from the same horizontal position. Note that, in both cases, the agent initially foveates the object in the first 20 time units, actively scans it for approximately the next 20 time units, and then either centers it in the case of the circle or avoids it in the case of the diamond. This foveate-scan-decide strategy was fairly typical of the evolved circle/diamond discriminators, although it was not universal. Even though the rays are uniformly distributed across the visual field, foveating the object still has the advantage of bringing the maximum number of rays to bear on the object and of placing the object in a standard position with respect to the agent.

It is important to emphasize that this agent is not merely centering and then statically pattern-matching an object. Rather, its strategy seems to be a dynamic one, with active scanning apparently playing an essential role. The importance of an agent having control of its own gaze direction has been a major theme in active vision research (Ballard, 1991; Churchland et al, 1994). Given that this task necessarily involves relative motion of the agent and the object (because the agent must express its decision by moving either toward or away from an object as it falls), it is perhaps not too surprising that dynamics appears to play an essential role in the operation of this controller.

The best circle/line discriminator had a mean fitness of 99.26% on the 24 evaluation trials and a mean fitness of 97.85% on 100 random trials. Qualitatively, all objects were correctly classified during both the 24 evaluation trials and the 100 random trials.

The behavior of this agent is shown in Figure 6, both while catching a circle (Figure 6a) and while avoiding a line (Figure 6b). The strategy here is rather different than for the circle/diamond discriminator described above. In this case, the agent initially *antifoveates* both the circle and the line, moving so that the object lies near the opposite periphery of its field of view. The agent then pauses and, as the object nears, the agent either centers it in the case of the circle or continues to move away in the case of a line. This antifoveate-and-decide strategy was fairly typical of the other circle/line discriminators that evolved, although it was not universal.

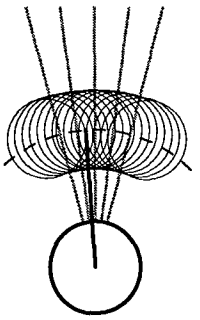


Figure 7: Experimental setup for pointing experiments. The agent is fixed in place and has an arm that swings along the dashed arc. Its rays are shown in gray. Overlapping circles denote the locations of the objects used to evaluate the agent's performance during evolution.

6 Pointing Experiments

Before a visually-guided agent can manipulate objects, it must be capable of coordinating the movement of its manipulator with objects appearing in its visual field. In a final set of experiments, stationary agents with a 1 degree-of-freedom transparent manipulator were evolved to point to the centers of circular objects appearing at arm's length in their visual field (Figure 7).

These agents had 5 rays and were unable to move. Instead, they had a transparent manipulator of length 45 with 1 angular degree of freedom and an angular range of $\pm\pi/4$. The angular velocity of the manipulator was proportional to the sum of two opposing torques (with a constant of proportionality of 0.1). Static circular objects of diameter 26 appeared at arm's length.

The performance measure to be maximized was the average angular accuracy of pointing, defined as:

$$1 - \frac{\sum_{i=1}^{NumTrials} \left| \theta_i^{object} - \theta_i^{arm} \right|}{MaxEvaluationAngle * NumTrials}$$

where θ_i^{object} is the angle of the object relative to the center of the body and θ_i^{arm} is the angle of the arm at the end of the i^{th} trial, *MaxEvaluationAngle* is the angular width over which objects can appear ($\pi/4$ in these experiments), and *NumTrials* is the total number of trials. A total of 15 evaluation trials were used during evolution, each of duration 20 time units, with the arm centered at the beginning of each trial.

To date, these pointing experiments have employed a bilaterally symmetric feedforward network with 5 ray sensory neurons projecting to 3 interneurons which in turn project to two motor neurons controlling the arm. In addition, the motor neurons received weighted inputs from two arm angle sensors (giving a total of 18 parameters). Arm angle sensors

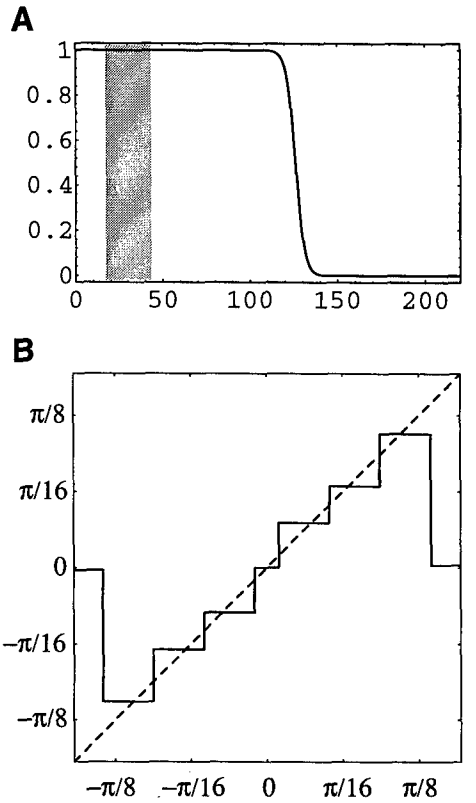


Figure 8: An evolved pointing agent. (a) A plot of the evolved ray sensor response (output vs. distance). The location of the target objects is shown in gray. (b) A plot of the final arm angle as a function of the angle of the target object. The response of a perfect pointer is shown as a dashed line.

gave a linear response from 0 (arm centered) to 1 (arm at an extreme angle on the side opposite the sensor). Time constants were fixed to 0.75.

Pointing agents with reasonably good accuracy were relatively easy to obtain by evolving population sizes as small as 50 for 25 generations with a mutation variance σ^2 of 5. The pointing behavior of a typical agent evolved with ray sensor bias $\in$ [-10,0] and ray sensor gain $\in$ [1,15] is shown in Figure 8b. This agent had a mean accuracy of 98.13% on the 15 evaluation trials and a mean accuracy of 98.06% on 100 random trials. As Figure 8b clearly shows, the pointing resolution of this agent is limited to a number of discrete angles. The reason for this is that this agent evolved a ray sensor response that is essentially binary in nature: each ray sensor gives a response near 1 if that ray intersects the object and a response near 0 if it does not (Figure 8a). This sort of response function was typical of agents evolved using wider ranges of ray sensor biases.

In order to achieve higher accuracy, we must give more careful consideration to the ray sensor biases and gains. Best results were obtained for a ray sensor response such as that shown in Figure 9a. Here, the ray sensor biases and gains have been chosen so that each ray sensor goes from an out-

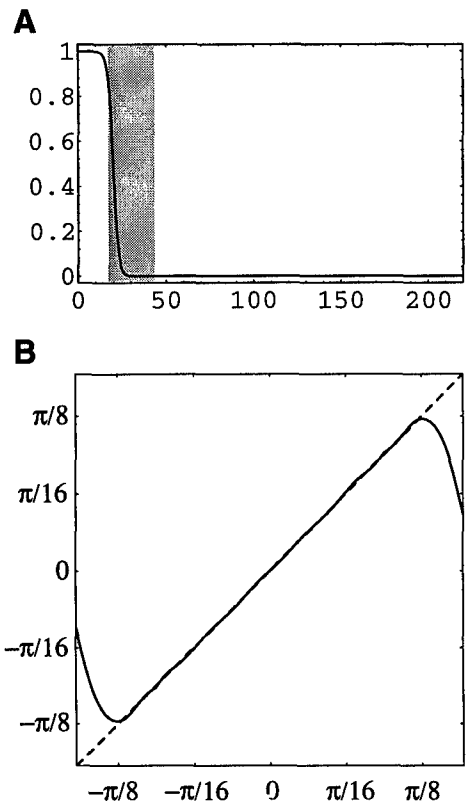


Figure 9: Another evolved pointing agent. (a) A plot of the response of a "near-optimal" ray sensor having a bias of -9.1 and a gain of 15. (b) A plot of the final arm angle as a function of the angle of a target object.

put of almost 0 near the center of the object to an output of almost 1 near the front, giving a maximum dynamic range. Best results were obtained by simply fixing the ray sensor biases and gains to these "near-optimal" values. The pointing behavior of one of the best agents evolved under these conditions is shown in Figure 9b (using a population size of 100 evolved for 200 generations with a σ^2 of 5). This agent had a mean accuracy of 99.87% on the 15 evaluation trials and a mean accuracy of 99.88% on 100 random trials. Note that this accuracy corresponds to an angular error of 0.054 degrees, which is very good considering the small number of rays and interneurons used.

7 Related Work

Perhaps the work most closely related to that described in the present paper is ongoing research at the University of Sussex on visually-guided behavior (Cliff, Harvey & Husbands, 1993; Harvey, Husbands & Cliff, 1994). This work also employs a class of dynamical neural network models and evolutionary algorithms to develop agents that can center themselves in an enclosed circular arena, visually orient to an object, track a moving object, and visually discriminate between two objects. However, the emphasis of this work has been on the evolution of neural controllers for actual visually-guided robots rather than the theoretical issues being explored here. Accordingly, they are concerned with the actual transmission and detection of light rays and they are working with a physical robot. In addition, their work to date has not considered visually-guided manipulation.

Broadly speaking, the COG project (Brooks & Stein, 1994), Brooks' attempt to apply his behavior-based robotics methodology to the construction of a humanoid robot head and torso, shares the goal of exploring the cognitive implications of adaptive behavior ideas. However, the COG project has not employed dynamical neural networks and evolutionary algorithms, and it does not especially emphasize dynamical ideas. On the other hand, the goals of the COG project are much more ambitious than those of the present work, and it is also using an actual physical robot.

8 Conclusions

This paper has sketched a visually-guided agent and demonstrated the evolution of dynamical neural networks for simplified versions of visually-guided orientation, object discrimination, and accurate pointing in this agent. The first two of these experiments also illustrated the importance of internal dynamics, which allows an agent's behavior to depend not only on its immediate circumstances, but also on its recent history of interaction with its environment. Regardless of how one chooses to define cognition, these results do represent a first step toward the evolution of more sophisticated agents. Future work will attempt to extend these capabilities in a variety of directions: selective orientation in the presence of multiple objects, discriminating among a larger set of objects, catching moving objects with an opaque hand, 2 degree-of-freedom movement and 3 degree-of-freedom manipulation, etc. Ultimately, these individual capabilities will need to be integrated into the complete agent sketched in Section 2.

However, it is worth noting that even the modest capabilities reported here already begin to raise some cognitively interesting questions. Consider, for example, the circle/diamond discrimination agents, which foveate and actively scan objects before catching or avoiding them. How do these agents achieve such a high accuracy with so few interneurons? Can we identify "circle" and "diamond" (or "smooth" and "pointy") detectors in these circuits? Will the notion of distributed representation that has been developed for static feedforward networks apply to agents controlled by dynamic recurrent circuits which actively control their perception? Or is it most appropriate to view these circuits as merely instantiating dynamics that, when coupled to the dynamics of their bodies and environments, give rise to effective performance of the tasks for which they were selected? Rather than debating competing intuitions in the abstract, experiments such as those described here pro-

vide concrete models within which such questions can be precisely framed and answered. Accordingly, as this research progresses, detailed studies of the operation of the circuits that evolve will be a major focus of attention.

Acknowledgments

I would like to thank Melanie Mitchell, Hillel Chiel, Tim van Gelder and Stewart Wilson for their comments on an earlier draft of this paper. This paper has also benefited from discussions with Andy Clark. This work was supported in part by the Santa Fe Institute, and in part by grant N00014-90-J-1545 from the Office of Naval Research.

References

Altman, J.S. & Kien, J. (1989). New models for motor control. *Neural Computation* **1**:173-183.

Bäck, T. & Schwefel, H. (1993). An overview of evolutionary algorithms for parameter optimization. *Evolutionary Computation* **1**(1): 1-23.

Ballard, D. (1991). Animate vision. *Artificial Intelligence* **48**:57-86.

Beer, R.D. (in press). The dynamics of adaptive behavior: A research program. To appear in *Robotics and Autonomous Systems*.

Beer, R.D. (1995a). A dynamical systems perspective on agent-environment interaction. *Artificial Intelligence* **72**:173-215.

Beer, R.D. (1995b). Computational and dynamical languages for autonomous agents. In (Port & van Gelder, 1995).

Beer, R.D. (1995c). On the dynamics of small continuous-time recurrent neural networks. *Adaptive Behavior* **3**(4):469-509.

Beer, R.D. & Gallagher, J.G. (1992). Evolving dynamical neural networks for adaptive behavior. *Adaptive Behavior* **1**(1):91-122.

Brooks, R. (1991). Intelligence without representation. *Artificial Intelligence* **47**:139-159.

Brooks, R. & Stein, L.A. (1994). Building brains for bodies. *Autonomous Robots* **1**:7-25.

Churchland, P.S., Ramachandran, V.S. & Sejnowski, T.J. (1994). A critique of pure vision. In C. Koch & J. Davies (Eds.), *Large-Scale Neuronal Theories of the Brain*. MIT Press.

Clark, A. & Toribio, J. (1994). Doing without representing? *Synthese* **101**:401-431.

Cliff, D., Harvey, I. & Husbands, P. (1993). Explorations in evolutionary robotics. *Adaptive Behavior* **2**(1):73-110.

Damasio, A. (1994). *Descartes' Error*. Putnam.

Gazzaniga, M.S., Ed. (1995). *The Cognitive Neurosciences*. MIT Press.

Goldberg, D.E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley.

Harvey, I., Husbands, P. & Cliff, D. (1994). Seeing the light: Artificial evolution, real vision. In Proceedings SAB94, pp. 392-401.

Haugeland, J. (1995). Mind embodied and embedded. In Yu-Houng & Jih-Ching Ho (Eds.), *Mind and Cognition* (pp. 3-38). Taipei, Taiwan: Academia Sinica.

Hutchins, E. (1995). *Cognition in the Wild*. MIT Press.

Knuth, D. (1981). *Seminumerical Algorithms*, 2nd Edition (The Art of Computer Programming, Vol. 2). Addison-Wesley.

Lakoff, G. (1987). *Women, Fire and Dangerous Things*. University of Chicago Press.

Port, R. & van Gelder, T. (1995). *Mind as Motion: Explorations in the Dynamics of Cognition*. MIT Press.

Posner, M.I. & Raichle, M.E. (1994). *Images of Mind*. W.H. Freeman.

Press, W.H., Teukolsky, S.A., Vetterling, W.T. & Flannery, B.P. (1994). *Numerical Recipes in C*, 2nd Edition. Cambridge University Press.

Rutkowska, J.C. (1994). Scaling up sensorimotor systems: Constraints from human infancy. *Adaptive Behavior* **2**(4):349-373.

Suchman, L. (1987). *Plans and Situated Actions*. Cambridge University Press.

Thelan, E. & Smith, L. (1994). *A Dynamic Systems Approach to the Development of Cognition and Action*. MIT Press.

Yamauchi, B. & Beer, R.D. (1994). Sequential behavior and learning in evolved dynamical neural networks. *Adaptive Behavior* **2**(3):219-246.

Emergence of a Multi-Agent Architecture and New Tactics For the Ant Colony Food Foraging Problem Using Genetic Programming

Forrest H Bennett III
Visiting Scholar
Computer Science Department
Stanford University
Stanford, California 94305 USA
EMAIL: fhb3@slip.net
PHONE: 415-424-0948

Abstract

Previous work in multi-agent systems has required the human designer to make up-front decisions about the multi-agent architecture, including the number of agents to employ and the specific tasks to be performed by each agent. This paper describes the automatic evolution of these decisions during a run of genetic programming using architecture-altering operations.

Genetic programming is extended to the discovery of multi-agent solutions for a central-place foraging problem for an ant colony. In this problem each individual ant is controlled by a set of agents, where agent is used in the sense of Minsky's Society of Mind.

Two new tactics for the central-place food foraging problem that were discovered by genetic programming are presented in this paper.

Genetic programming was able to evolve time-efficient solutions to this problem by distributing the functions and terminals across successively more agents in such a way as to reduce the maximum number of functions executed per agent. The other source of time-efficiency in the evolved solution was the cooperation that emerged among the ants in the ant colony.

1 Introduction

The thesis that minds are organized into *agents* has been presented in the literature of several different fields. Minsky (1985) presents the following notion of agents:

> "I'll call 'Society of Mind' this scheme in which each mind is made of many smaller processes. These we'll call *agents*. Each mental agent by itself can only do some simple thing that needs no mind or thought at all. Yet when we join these agents in societies -- in certain very special ways -- this leads to true intelligence."
>
> "To explain the mind, we have to show how minds are built from mindless stuff, from parts that are much smaller and simpler than anything we'd consider smart. Unless we can explain the mind in terms of things that have no thoughts or feelings of their own, we'll only have gone around in circles."

In cognitive science, Dennett 1991 argues that "there is no central Headquarters" in the mind, but instead says that "specialist circuits" operate in "parallel pandemoniums."

In psychology, Baars 1988 summarizes a "gathering consensus" in his field that minds are implemented as a "distributed society of specialists."

In this paper we use the term *agents* in the sense of these references -- that is, the narrow specialist modules that operate in parallel to create an effective overall problem solving organism.

Herein, an agent corresponds to a single branch of the program tree for an individual in the population of the genetic programming system.

Are the minds of real ants organized into agents? This is not known. However, it is known that a typical ant species has as many as 24 to 40 different categories of behaviors; and some species seem to have as many as 50 categories of behaviors (Oster and Wilson 1978). It is easy to envision that each of these behaviors corresponds to one or more agents.

Multi-agent systems are also attractive because of the potential gain in efficiency from their inherent parallelism.

1.1 Collective Behavior in Ant Colonies

Social insects such as ants, wasps, bees, and termites dominate the insect fauna and together constitute more than 75% of the insect biomass (Fittkau and Klinge 1973). The success of these insects is due in large part to the survival advantage accruing from their cooperation in a complex social organization and to the efficiencies of specialization that this cooperation makes possible. Of the social insects, ants have the most complex social organizations (Holldobler and Wilson 1990).

The problem presented in this paper is a single aspect of an ant colony's cooperative behavior, namely an abstraction of the central-place foraging problem faced by

worker ants. In the version of the problem presented here, as in nature, a single worker ant can perform food foraging competently (Holldobler and Wilson 1990). However, cooperation yields more efficient foraging.

Food sources are initially located by individual *Solenopsis invicta* ants using random search (Holldobler and Wilson 1990). Once a food source is located, the ants lay an alpha-farnesene recruitment pheromone on the ground while returning to the nest, thereby creating a trail of pheromones connecting the food source and the nest. When the odor of this pheromone is detected by other worker ants, they respond by following the pheromone trail. Thus, other ants in the colony are able to go much more directly to a food source that has already been found without having to search for the food. The pheromone-dropping behavior together with the pheromone trail-following behavior is how the colony cooperates to perform food foraging (Beckers et al. 1990). The cooperative behavior is accomplished with each ant responding only to locally available information.

In this paper, the behavior of each individual ant is controlled by a multi-agent algorithm (multi-branch program) where each agent (branch) is executed in parallel.

A solution to an ant colony food foraging problem where each ant is running a single sequential algorithm was written by Resnick (1991), and another sequential algorithm was evolved using genetic programming in Koza 1992 (Chapter 12). A collective solution for a group of robots to a similar object clustering problem has been studied (Deneubourg et al. 1990) and implemented using physical robots (Beckers, Holland, and Deneubourg 1994).

The problem in this paper is a harder version of the problem described in Koza 1992 because the function and terminal set used by Koza was much more powerful than the primitives used here (see sections 4.1 and 4.2).

1.2 Background on Genetic Programming

Genetic Programming (Koza 1992) is an extension of the genetic algorithm (Goldberg 1989) in which the individuals in the population consist of computer programs represented as expression trees.

A genetic programming run begins by creating an initial population (generation zero) of program trees that have random shapes and sizes. The nodes in the program trees are functions and terminals appropriate for a given problem. The nodes in each initial tree are randomly chosen from the set of functions and terminals being used. Each internal node of the program tree is a function and has a child node for each of its arguments. Each leaf node of the program tree is a terminal.

In each generation, the genetic operators of crossover, reproduction, and mutation are applied to the individuals (program trees) in the population in order to create a new population of individuals. The genetic operators are applied fitness proportionally. Therefore individuals that are more fit produce more progeny in the next generation.

The fitness of each individual in the population is assigned by executing that individual and measuring how well it performs at the problem one is trying to solve.

The evolutionary process proceeds generation by generation until a solution to the problem is found, or the maximum number of allowed generations is reached.

1.3 Previous Genetic Programming Multi-Agent Work

Previous work in multi-agent systems has required the human designer to make up-front decisions about the multi-agent architecture, including the number of agents to employ and the specific tasks to be performed by each agent. This paper describes the automatic evolution of these multi-agent architectures during a run of genetic programming using architecture-altering operations.

In Andre 1995, agents were evolved where the number of agents (two) and the task of each agent was predetermined. In Ryan 1995, the conditions under which each agent would activate was preset, as were the number of agents. In Haynes, et al 1995 the number of agents was preset to four. These were not Minsky-style agents, but rather members of a cooperating team that each controlled its own body. Each agent in Crosbie and Spafford 1995 was generated in a separate run of genetic programming; they were not co-evolved.

1.4 Previous Genetic Programming Work with New Programming Elements

Genetic programming has previously been extended to automatic definition and manipulation of well known programming elements while simultaneously solving a problem. Examples include hierarchical modules (Angeline and Pollack 1992), subroutines with parameters (Koza 1994), program architecture (Koza 1995), generic control structures such as macros (Spector 1996), and iteration (Koza and Andre 1996).

This paper extends genetic programming to the automatic definition and manipulation of parallel algorithms and parallel architectures.

2 The Foraging Problem

Genetic programming is applied to the problem of finding a multi-agent parallel algorithm that, when executed by each ant in a colony of 20 ants, causes efficient central-place foraging behavior in the ant colony. We seek to evolve the ant colony's collective behavior rather than to design it as in Mataric (1993) and Beckers (1994). The behavior of each individual ant in this problem is controlled by multiple agents.

The fitness function used in this problem penalizes the

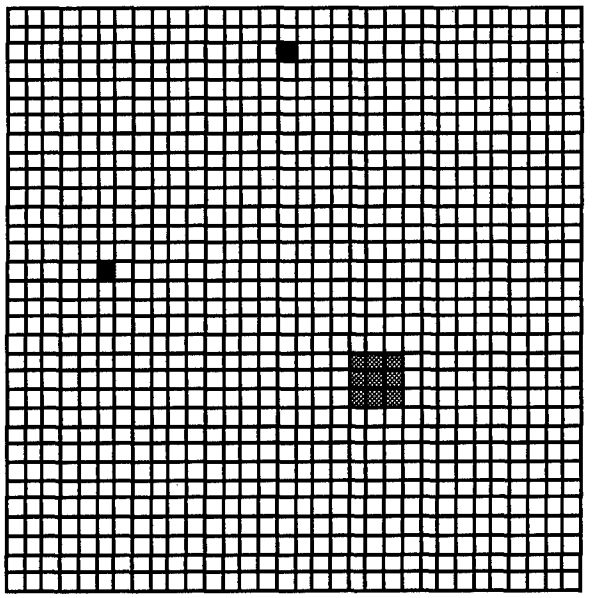


Figure 1: The 32 by 32 grid for the toroidal ant colony world. The two food locations with 72 food pellets each are shown in black. The nine grid locations of the nest are shown in gray. (Location (0, 0) is in the lower left corner.)

ant colony for the time it uses to find and transport each food pellet to the nest and penalizes the colony heavily for each food pellet it does not transport to the nest. Thus there is selection pressure for transporting all the food and for doing so efficiently. A colony in which each ant randomly searches for each food pellet will take a long time to gather the food and thus be less fit. The ant colony can be more competitive by discovering how to cooperate.

The second way that the ant colony can become more efficient (and thus more competitive) is by distributing the functions and terminals in the program tree into as many agents as possible, while still solving the problem. The time-efficiency of a multi-agent algorithm is measured in terms of the greatest number of functions executed in any one of the agents. Thus an algorithm that executes 100 functions in 4 agents (with 25 functions per agent) will be twice as time-efficient as an algorithm that executes 100 functions in 2 agents (with 50 functions per agent). This will be discussed in greater detail in section 4.3.

2.1 The Simulated World

Each of the 20 ants in the colony are controlled by the same multi-agent algorithm (i.e., an individual multi-branch program tree in the genetic programming population).

The colony's world is a 32 by 32 toroidal grid as shown in figure 1. Each of the colony's ants begins each fitness evaluation at a random location within 5 squares of the center of a 3 by 3 nest that is centered at location (21, 11).

Each ant is initially facing in a random direction. There are a total of 144 food pellets located in two separate grid locations that each have 72 pellets. One food area is at (5, 17) and the other at (15, 29).

The ant colony is simulated until either all the food pellets have been transported to the nest, or the maximum number of allotted time steps (4,000) is reached.

On each time step of the simulation, the evolved multi-agent algorithm for the colony is evaluated once for each of the 20 ants in the colony. Evaluating the multi-agent algorithm for a single ant involves evaluating each agent of the algorithm in a manner that simulates parallel execution.

The result of each agent evaluation is a single action (designated by a terminal) to be performed in the world. The action (result terminal) of each agent is not executed until all of the agents have been evaluated. This delay of the actions insures that all the functions (conditional tests) are evaluated as if they were in parallel, with the world and the ant in the same state. Thus the order of the agents (branches) in the overall program tree is irrelevant.

After each of the agents (branches) of the program tree have been evaluated, the designated terminal actions are executed to change the state of the ant and the world. Each specific behavior (corresponding to a terminal) is only executed one time per ant per time step even if more than one agent evaluates to that action. Regardless of the order of the agents, the actions are always executed in the following order: `GRAB-FOOD`, `UNCONDITIONAL-DROP-PHEROMONE`, `TURN-LEFT`, `TURN-RIGHT`, `MOVE-RANDOM`, and `MOVE-FORWARD`. The two movement actions are executed last to insure that all the actions take place in the ant's current location before any movement occurs.

Figure 2 shows an example program tree with three agents. Suppose this program tree is evaluated for ant on a grid location containing food and that the ant is not currently carrying food. Since `IF-FOOD-HERE` is true the result of the first (leftmost) agent will be the `GRAB-FOOD` terminal. The result of the second (middle) agent will always be the `MOVE-FORWARD` terminal. The result of the third (rightmost) agent will be the `MOVE-RANDOM` terminal, since the ant is not currently carrying food. So the results of evaluating all the agents is the set of terminals {`GRAB-FOOD`, `MOVE-FORWARD`, `MOVE-RANDOM`}. Now that we have the terminals from each of the agents, the three actions are executed. First, `GRAB_FOOD` removes one food pellet from the current grid location and sets the state of the ant to indicate that it is carrying food. Second, the `MOVE_FORWARD` changes the ant's grid location by one square in the direction the ant is currently facing. Third, the `MOVE-RANDOM` is executed. (See the details on the set of terminals below.)

3 Architecture-Altering Operations Used to Create the Multi-Agent Architecture

3.1 Branch Duplication Operation

Branch duplication involves creating a duplicate copy of one

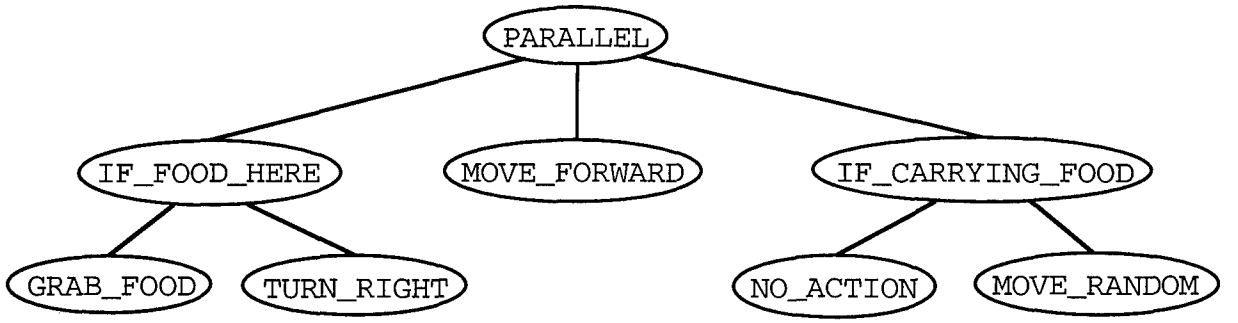


Figure 2: An example program tree showing an individual with three agents. Each agent is represented by one of the three branches connected by the `PARALLEL function`.

of the branches (agents) in an individual multiple agent parallel program. This operation was inspired by naturally occurring gene duplication (Ohno 1970).

The branch duplication operation used here is a simpler version of the one presented in Koza (1995). Koza's version of branch duplication takes into account modifications to hierarchically referenced branches. However, the branches used for this problem do not have any hierarchical references. Thus the steps involved to handle these references during branch duplication are not required.

This operation is applied to 3% of the population in each generation. The branch duplication operation used here works in the following way:

(1) Select an individual from the population based on fitness using tournament selection (tournament size of 7).

(2) Randomly pick one of the branches in this individual as the branch-to-be-duplicated.

(3) For the branch picked, make a duplicate copy of that branch, and insert it into the individual. This increases by one the number of branches in the individual. If the individual selected already has the maximum number of allowed branches, then do nothing.

Initially this new duplicate branch has no effect, since it does the same thing as the branch from which it was copied. However, the two branches may diverge in future generations because of crossover or mutation.

3.2 Branch Deletion Operation

Branch deletion involves deleting one of the branches (agents) in an individual multiple agent parallel program.

The branch deletion operation used here is also simpler than Koza's because there are no hierarchical references.

This operation is applied to 1% of the population in each generation. The branch deletion operation used here works in the following way:

(1) Select an individual from the population the same way as we did for branch duplication (previous section).

(2) Randomly pick one of the branches in this individual as the branch-to-be-deleted.

(3) For the branch picked, delete it from the selected individual. This decreases by one the number of branches in the individual. If the individual selected has only a single branch, then do nothing.

This operation will affect the behavior of the individual unless the deleted branch is a duplicate copy of another branch, the deleted branch was not doing anything, or the effects of the deleted branch are the same as the effects of some combination of one or more of the other branches.

4 Preparatory Steps for the Genetic Programming Run

All the programs in the initial population of generation 0 are created with a single branch (agent). Thus the run begins with all programs in the initial population having the minimum architectural structure.

All the functions used in this problem correspond to percepts and all the terminals used correspond to actions that change the state of the world or the ant.

4.1 Function Set

All the functions in this problem are two-argument conditional branches that have no side effects. The function set consists of the eight functions that are all available to every branch (agent). The effects of evaluating the various functions are as follows:

- `IF-FOOD-HERE` is a two-argument conditional branching function that executes its first branch if there is a food pellet at the same location on the grid as the ant, but otherwise executes its second branch.
- `IF-FOOD-FORWARD` is a two-argument conditional branching function that executes its first branch if there is a food pellet in the grid location adjacent to the ant's location in the direction the ant is facing, but otherwise executes its second branch.
- `IF-CARRYING-FOOD` is a two-argument conditional branching function that executes its

first branch if the ant is carrying a food pellet, but otherwise executes its second branch. This conditional will be true if the ant has grabbed a food pellet and the ant has not yet returned to the nest.

- IF-SMELL-FOOD is a two-argument conditional branching function that executes its first branch if there is at least one food pellet on one or more of the four (north, east, south, and west) grid locations adjacent to the ant's current location, but otherwise executes its second branch.
- IF-NEST-HERE is a two-argument conditional branching function that executes its first branch if the ant is currently in a grid location that is one of the grid locations of the nest, but otherwise executes its second branch. When an ant carrying food enters the nest, that food pellet is implicitly taken from the ant and dropped into the nest.
- IF-FACING-NEST is a two-argument conditional branching function that executes its first branch if the ant is currently facing in the direction of the nest, but otherwise executes its second branch. The ant is considered to be facing in the direction of the nest if and only if the execution of a MOVE-FORWARD terminal would not increase the ant's manhattan distance from the center of the nest. Real ants are able to navigate back to the nest using the polarized light in the sky (Holldobler and Wilson 1990).
- IF-SMELL-PHEROMONE is a two-argument conditional branching function that executes its first branch if there is at least one pheromone on one or more of the four (north, east, south, and west) grid locations adjacent to the ant's current location, but otherwise executes its second branch.
- IF-PHEROMONE-FORWARD is a two-argument conditional branching function that executes its first branch if there is a pheromone in the grid location adjacent to the ant's location in the direction the ant is facing, but otherwise executes its second branch.

The function set used by Koza (1992) in his version of the central-place foraging problem for an ant colony was powerful. Indeed, some of the built-in functions provided to the genetic programming system were already as complex as the kind of agents that we wanted to evolve in the experiment described here. For example, Koza's function MOVE-TO-ADJACENT-FOOD-ELSE combines a test of all the adjacent positions in the ants environment, with a move into the location with the food pellet in it. Thus the evolutionary process did not have to learn which location contained the food pellet, and did not have to link this information with the action of moving into that location. Most importantly, it did not have to discover that linking that information with that behavior was a useful thing to do for this problem. In the version of the problem presented here, genetic programming has to discover an equivalent to Koza's MOVE-TO-ADJACENT-FOOD-ELSE function by using some combination of: IF-SMELL-FOOD, TURN-LEFT, TURN-RIGHT, MOVE-RANDOM, IF-FOOD-FORWARD, and MOVE-FORWARD. The same point applies to Koza's rather powerful function MOVE-TO-ADJACENT-PHEROMONE-ELSE.

Koza also uses a PROGN function that allows a list of arguments to be evaluated in series. This function, in effect, provides for a kind of state information to be used. For example, in the branch (PROGN (MOVE-FORWARD) (TURN-LEFT)), the branch is able to execute a TURN-LEFT in the state in which a MOVE-FORWARD has just occurred. Omitting PROGN prevents the use of implicit state information, and prevents solutions that merely encode a route to the food with a sequence of moves and turns.

4.2 Terminal Set

All the terminals in this problem are actions that have effects on the state of the world or the ant. The terminal set consists of the following seven terminals that are all available to every branch (agent). The effects of evaluating the various terminals are as follows:

- MOVE-FORWARD changes the ant's location to the adjacent grid location in the direction that the ant is currently facing.
- TURN-RIGHT changes the direction the ant is facing by 90 degrees clockwise.
- TURN-LEFT changes the direction the ant is facing by 90 degrees counter-clockwise.
- MOVE-RANDOM changes the direction the ant is facing to one of the four possible directions (north, east, south, and west) with equal probability and then moves the ant two squares forward in that direction.
- GRAB-FOOD takes one food pellet from the grid location where the ant is currently located, if and only if there is a food pellet at that location and the ant is no already carrying a food pellet.
- UNCONDITIONAL-DROP-PHEROMONE drops a pheromone cloud into a 3 by 3 square centered at the current location of the ant. The pheromones decay linearly such that they are no longer present on the grid locations 20 simulation time steps after they are dropped.
- NO-ACTION does nothing, and was included so that an agent could decide that no action was appropriate for it in a given time step based on the percepts it had evaluated.

The DROP-PHEROMONE terminal used in Koza 1992

is a powerful operator that only executes when the ant is carrying food. Again, the discovery of the utility of only using the DROP-PHEROMONE terminal under certain conditions is an example of the kind of agent that we wanted to evolve. The UNCONDITIONAL-DROP-PHEROMONE terminal used here always drops pheromones.

The MOVE-TO-NEST terminal used by Koza combines the perception of the direction of the nest with the action of movement towards the nest. Again, we seek to evolve an agent that combines the percept and the action. This experiment uses movement terminals that are separate from the IF-FACING-NEST percept function.

4.3 Fitness and Hits

The colony's hits are the number of food pellets that it transports to the nest. The maximum hits is 144.

The fitness measure combines the amount of time used by the colony to transport food to the nest and the number of functions executed to do so; and it includes a large penalty for each food pellet not transported to the nest. The fitness formula is

$$\frac{\Sigma^{n}(t_{food} * f_{food}) + \Sigma^{m}(t_{max} * f_{max} * d_{food})}{1{,}000{,}000} \quad (1)$$

where n is the number of food pellets transported to the nest, t_{food} is the number of time steps elapsed when the food pellet arrived at the nest, f_{food} is the number of sequential IF functions (as defined below) executed by the ant who transported the food pellet, m is the number of food pellets not transported to nest, t_{max} is the maximum allotted time steps (4,000), f_{max} is the maximum possible value of f_{food} (t_{max} * p_{max} or 400,000), d_{food} is the manhattan distance between food pellet and nest, and p_{max} is the maximum number of points per agent (100).

The number of sequential IF functions executed by an ant to transport a food pellet to the nest is determined as follows. When the agents in the program tree are evaluated for a single time step for a single ant, the maximum number of functions evaluated by one of the agents is counted as the number of sequential IF functions executed by that ant for that time step. A running total of this count is stored for each ant, and the count for an ant is incorporated into the fitness as noted above and then cleared each time the ant arrives at the nest with a food pellet.

Thus, any food pellets not transported to the nest are penalized as though they were brought the remaining distance to the nest by the largest size agent possible that also takes the maximum number of time steps to transport the food to the nest.

The colony's fitness is calculated for a single fitness case, that is for a single world configuration of nest location, food pellet locations, and total number of food pellets. However, note that each fitness evaluation is in two minor senses a distinct fitness case. First, the successive executions of the MOVE-RANDOM terminal will move in different directions in different fitness evaluations. Second, the starting positions of the ants and the directions the ants are facing are different each time the colony is executed. Therefore, an algorithm could not rely on a particular sequence of MOVE-RANDOM executions across generations.

Even though the food is located in the same places in each fitness evaluation, the colony has no way with the given function and terminal set to encode the location of the food (not even implicitly). Therefore, the colony must actually search for the food in each fitness evaluation.

4.4 Other Parameters for the Run

The population size, M, was set to 64,000.

The percentages for the genetic operations were: 75% for internal point crossover, 10% for leaf point crossover, 10% for reproduction, 3% for branch duplication, 1% for branch deletion, and 1% for mutation.

A maximum size of 100 points was allowed for each agent. A maximum of 7 agents were allowed per individual, so the maximum size of an individual was 700 points.

The other parameters for controlling the runs were the default values specified in Koza (1994).

4.5 Termination Criterion and Results Designation

There was no termination criterion for this problem since a fitness of zero was not possible, and there was no way of knowing in advance how good of a fitness might evolve. The run was terminated manually after the fitness value of the best-of-run individual had appeared to plateau.

4.6 Parallel Computer System

The hardware used for this problem is a 64 node parallel computer where each node consists of an 80 MHz PowerPC 601 microprocessor with 32M bytes of RAM. The inter-process communication is handled by an INMOS Transputer at each node and runs at 10M bits/sec. The 64 nodes are connected in a toroidal grid so that each node communicates only with its four nearest neighbors.

Each node runs a distinct copy of the genetic programming system with a population of 10,000 individuals. On each generation, 200 individuals are selected based on fitness to migrate to each of the four neighboring nodes. See Andre and Koza 1996.

The four distinct levels of parallelism employed in this problem are: the physical parallelism of the 64 processors

of the actual hardware used, the conceptual parallelism of the individuals in the population of programs in the genetic programming system, the simulated parallelism of the 20 ants within one ant colony, and the simulated parallelism of the multiple agents controlling a single ant.

5 Results

5.1 Results from Intermediate Generations

The best individual of generation 0 has a fitness value of 4,258,013, scored 26 hits, and has a single agent. (All the individuals in generation 0 have a single agent.) The effect of this agent is to wander randomly until it finds food, grab any food that it finds, and randomly wander back to the nest. It does not use smell, orient to the nest, or cooperate in any way with the other ants in the colony. The program tree for this individual is shown here in simplified form:

```
(IF_FOOD_HERE
   (IF_CARRYING_FOOD
      (MOVE_FORWARD) (GRAB_FOOD))
   (MOVE_RANDOM)).
```

The first best-of-generation individual to score all 144 possible hits occurred in generation 7, had a fitness value of 568, and contained two agents. This individual has 79 points in the first agent and 93 points in the second agent.

The best fitness drops by 3 orders of magnitude in generation 7 (see Figure 3). This large drop in fitness is because the best-of-generation colony in generation 7 transported all the food to the nest, and thus does not incur the large fitness penalty for un-transported food. From this generation forward, all the best-of-generation individuals score 144 hits. However, the best fitness continues to decrease (or remain the same) in every generation because the best-of-generation individuals were able to get more efficient between generations 7 and 90. Figure 3 shows that between generations 7 and 90, the number of agents in the best-of-generation individual increases as the best-of-generation fitness decreases.

In generation 87, the best-of-generation individual had 7 agents. This was the only generation in which the best individual had the maximum number of agents.

5.1 Best-of-run Individual

The best individual of the run appeared in generation 90, had a fitness value of 7.4, and scored 144 hits. This individual has 6 agents, with a total of 380 points (i.e., functions and terminals) in all 6 agents combined.

When all 6 agents in the best-of-run individual are run together and viewed in an animation, one can observe how the colony solves the problem. First, each ant wanders randomly until it finds food, whereupon it grabs a food pellet and proceeds directly back to the nest while dropping pheromones along the way. Other randomly wandering ants that come upon this pheromone trail follow it to the food and join the food carrying and pheromone dropping. The recruitment, carrying, and pheromone dropping continue until the food source is exhausted. Then the ants resume random wandering and the pheromone trail drys up. Sometimes the second food source is discovered while the first food source is still being exploited. In this case, the colony works on both food sources at the same time, and when one of the food sources is exhausted all of the recruited ants focus on the remaining food source.

In order to determine the behavior of each of the agents in the best-of-run individual, an animation of the fitness evaluation was run where a single agent was allowed to control one half of the ants in the colony, while the ants in the other half of the colony were controlled by all six agents. The qualitative behavior of ants controlled only by a single agent are as follows:

- *Agent 0:* The ant only moves in a straight line in the direction that it was initially facing, and doesn't pick up any food or drop pheromones.
- *Agent 1:* The ant wanders randomly until it finds either a pheromone or food, then it stops next to that pheromone or food and does nothing else. When the pheromone the ant is next to decays or the food pellet the ant is next to is carried away, it resumes random wandering.
- *Agent 2:* When the ant happens upon the nest it stops there and does nothing else. Otherwise, it only moves in a straight line in the direction that it was initially facing, and doesn't pick up any food or drop pheromones.
- *Agent 3:* The ant behaves like it does when under the control of agent 1 except that the ant stops on top of the food instead of next to it.
- *Agent 4:* The ant moves to the corner of the world and stops there and does nothing else. However, if it encounters a food pellet on the way to the corner it stops on top of that food pellet and stays there until the food pellet is carried away, whereupon the ant moves to the corner of the world and stops.
- *Agent 5:* Behaves the same as agent 3.

When both agent 0 and agent 1 are run together to control the colony, the ants perform random foraging, and grab the food and transport it back to the nest, but do not drop pheromones or cooperate in any way.

When the three agents 0, 1, and 2 are all run together to control the colony, the ants perform the full cooperative pheromone-dropping and pheromone-trail-following behavior, but still do not perform as efficiently as all 6 of the agents running together.

When half the ants in the colony are controlled only by agents 2 and 3, and the other half is controlled by all 6 agents, an interesting new tactic was observed. The ants

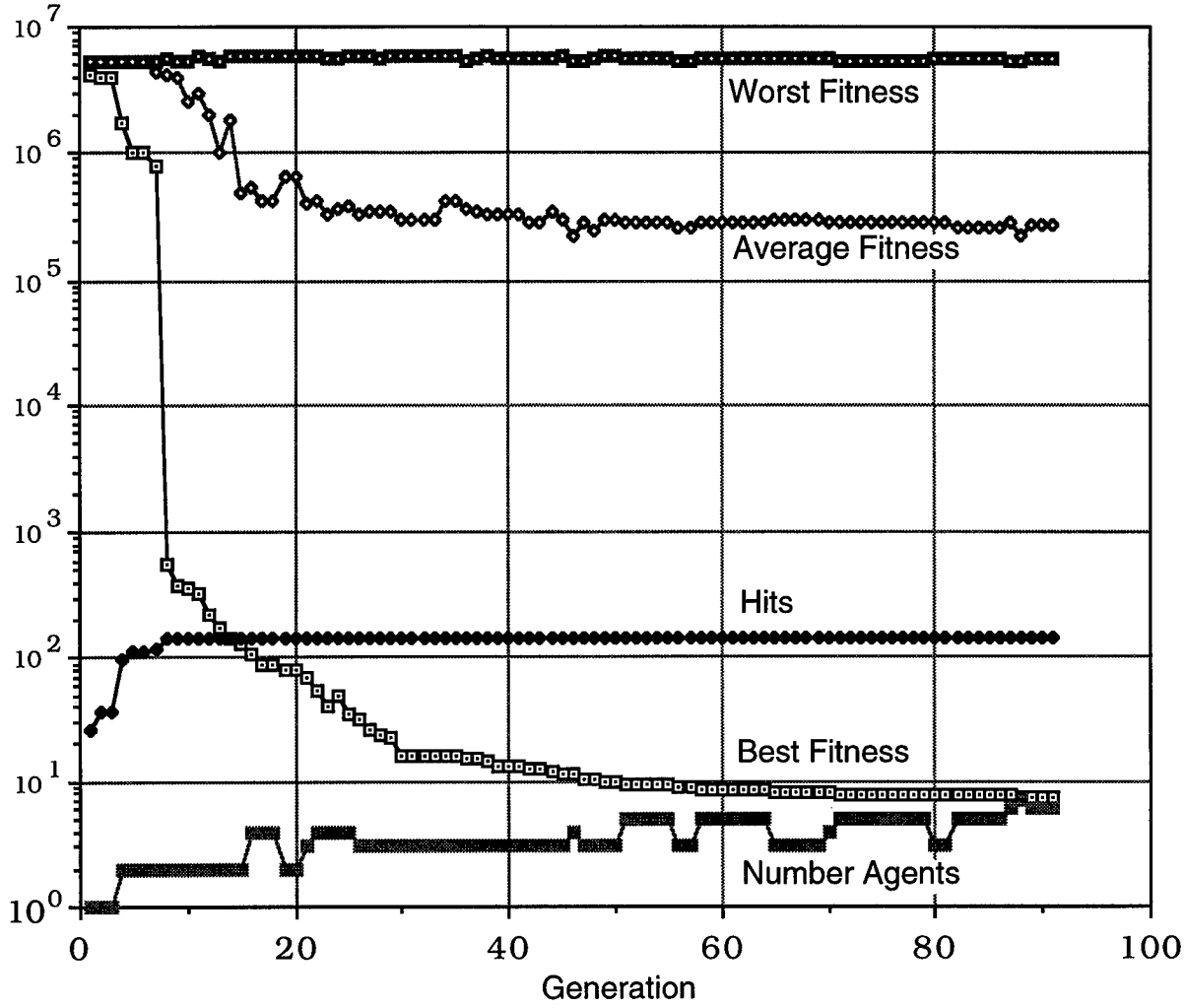


Figure 3: Log plot by generation of Worst-fitness, Average-fitness, Hits for best-of-generation, Best-fitness, and Number of agents for. best-of-generation.

controlled only by agents 2 and 3 would wander randomly until they either found a pheromone trail and followed it to the food, or until they came directly upon food. Once they arrived at food, they would only carry it back to the nest if there were no pheromones around the food, otherwise they would just stop at the food. This behavior had two consequences for food foraging. If an ant controlled only by agents 2 and 3 was the first to find food (thus there would be no pheromone trail around the food), it would carry the food back to the nest and lay the pheromone trail as expected. However, if one of these ants found food that already had pheromones around it (and was thus already being exploited), the ant would just stop there and wait. Then, if the pheromones around the food decayed completely, any ant sitting there at the food would then pick the food up and proceed back to the nest while rebuilding a new pheromone trail. Also, if the food pile where the ant is waiting becomes exhausted, the ant will resume random searching.

Thus, ants controlled only by agents 2 and 3 will build a new pheromone trail to a food pile when a pheromone trail drys up due to lack of recruitment. The benefit of this tactic is that the food does not have to be discovered again by random searching. The cost of this tactic is that these waiting ants are not helping to transport food.

This "waiting" tactic does not happen when all 6 agents are controlling an ant. However, this tactic could have been part of the behavior of an ancestor of the best-of-run individual. Furthermore, if an experiment were run where the environment (distance to food, pheromone decay rate, number of ants in the colony, size of the world, etc) of the colony changed, selection pressure could act to cause this behavior to be expressed and result in a fitness advantage.

The behavior of stopping at the nest that was observed in ants controlled only by agent 2 could potentially be part of a useful tactic under certain circumstances as well. Under the right combination of world size, size of the colony, pheromone decay rate, distance to food, etc, having some ants wait by the nest for the recruitment trail could be more efficient than having all the recruited ants discover the

recruitment trail randomly. In fact, having most of the ants wait at the nest to be recruited corresponds more to the behavior of real ants.

The best individual was tested 10,000 times to determine how well it would generalize over many different random sequences of MOVE-RANDOM. For each of these 10,000 trials this individual scored the maximum possible of 144 hits. The average fitness in these 10,000 trials was 14.6 (compared to 7.4 during the run) and the standard deviation was 6.3.

This run produced many individuals that scored the maximum possible 144 hits. Individuals that scored 144 hits were found that contained every number of agents from two to the maximum allowed (seven).

The evolved code for the four of the six agents in the best-of-run individual is given here (agents 3 and 5 are omitted since they are almost the same as agent 1):

```
Agent 0: (if-facing-nest (if-facing-nest (move-
forward) (turn-left)) (if-smell-food (if-food-
forward (if-nest-here (if-facing-nest (move-
forward) (turn-left)) (if-food-here (drop-pher)
(move-forward))) (if-nest-here (move-forward)
(turn-left))) (move-forward)))
```

```
Agent 1: (if-carrying-food (if-facing-nest (if-
carrying-food (if-smell-pher (drop-pher) (if-
facing-nest (if-facing-nest (if-food-here (move-
forward) (move-forward)) (turn-random)) (turn-
right))) (if-facing-nest (move-forward) (turn-
left))) (if-facing-nest (if-food-forward (move-
forward) (turn-right)) (turn-left))) (if-smell-
food (if-food-forward (if-nest-here (if-
carrying-food (if-facing-nest (if-facing-nest
(move-forward) (turn-left)) (if-pher-forward
(grab-food) (move-forward))) (if-food-here
(turn-right) (if-facing-nest (move-forward)
(turn-right)))) (if-carrying-food (if-facing-
nest (if-smell-pher (grab-food) (grab-food))
(if-pher-forward (grab-food) (move-forward)))
(if-food-here (if-food-here (grab-food) (drop-
pher)) (if-food-forward (move-forward) (if-food-
here (turn-right) (turn-random))))))) (if-nest-
here (move-forward) (turn-left))) (if-pher-
forward (grab-food) (if-facing-nest (if-
carrying-food (turn-right) (turn-random)) (if-
smell-pher (turn-left) (turn-random))))))
```

```
Agent 2: (if-carrying-food (if-facing-nest (if-
nest-here (if-nest-here (if-facing-nest (turn-
right) (if-smell-pher (turn-left) (no-action)))
(move-forward)) (if-smell-pher (move-forward)
(if-smell-pher (turn-left) (drop-pher)))) (if-
nest-here (grab-food) (if-smell-pher (if-facing-
nest (turn-right) (if-smell-pher (turn-left)
(no-action))) (if-smell-food (no-action) (if-
food-forward (move-forward) (move-forward))))))
(if-nest-here (if-carrying-food (if-food-here
(if-facing-nest (no-action) (if-nest-here (turn-
random) (move-forward))) (if-pher-forward (grab-
food) (no-action))) (if-nest-here (if-smell-food
(if-food-here (if-facing-nest (no-action) (if-
nest-here (turn-random) (move-forward))) (if-
pher-forward (grab-food) (no-action))) (if-food-
here (if-facing-nest (if-carrying-food (grab-
food) (grab-food)) (if-facing-nest (if-food-here
(move-forward) (move-forward)) (turn-random)))
(if-carrying-food (grab-food) (grab-food))))
(if-smell-food (grab-food) (move-forward))))
(if-smell-food (grab-food) (move-forward))))
```

```
Agent 4: (if-facing-nest (if-carrying-food (if-
facing-nest (if-facing-nest (move-forward)
(turn-left)) (if-pher-forward (grab-food) (move-
forward))) (if-food-forward (if-nest-here (if-
carrying-food (if-facing-nest (if-facing-nest
(move-forward) (turn-left)) (if-pher-forward
(grab-food) (move-forward))) (if-food-here (if-
food-here (grab-food) (drop-pher)) (if-food-
forward (if-nest-here (if-pher-forward (grab-
food) (move-forward)) (turn-left)) (turn-
right)))) (if-food-here (drop-pher) (move-
forward))) (if-nest-here (if-carrying-food (if-
facing-nest (if-facing-nest (move-forward)
(turn-left)) (if-pher-forward (if-food-forward
(if-nest-here (if-food-here (if-facing-nest
(turn-left) (grab-food)) (if-smell-pher (turn-
random) (if-nest-here (drop-pher) (turn-left))))
(if-food-forward (if-food-here (turn-right) (no-
action)) (grab-food))) (if-nest-here (turn-left)
(move-forward))) (move-forward))) (if-food-here
(if-food-here (grab-food) (drop-pher)) (if-food-
forward (if-nest-here (move-forward) (turn-
left)) (turn-right)))) (turn-left)))) (if-smell-
food (if-food-forward (if-food-here (if-food-
here (grab-food) (drop-pher)) (if-food-forward
(move-forward) (if-food-here (turn-right) (if-
food-forward (if-nest-here (if-facing-nest (if-
facing-nest (move-forward) (turn-left)) (if-
food-forward (move-forward) (if-food-here (turn-
right) (turn-random)))) (move-forward)) (if-
nest-here (move-forward) (turn-left))))))) (if-
nest-here (move-forward) (turn-left))) (move-
forward)))
```

6 Conclusions

Genetic programming was successfully used to discover multi-agent algorithms that solve a central-place foraging problem for an ant colony. Genetic programming was able to simultaneously evolve time-efficient multi-agent algorithms, multi-agent architectures, and emergent cooperation within the colony of ants.

Two new tactics for the central-place food foraging problem were discovered by genetic programming.

Time-efficient solutions to this problem were evolved by the emergence of cooperation and by distributing the

functions and terminals across successively more agents in such a way as to reduce the maximum number of functions executed per agent.

Acknowledgements

The author gratefully acknowledges David Andre for the use of his genetic programming code and for his help in using it, John Koza for his encouragement to undertake this project, Mark Roulo for his discussions during the project, and Simon Handley and Scott Brave for reading and critiquing drafts of this paper.

Bibliography

Andre, David. 1995. The evolution of agents that build mental models and create simple plans. In Eshelman, Larry J. (editor). *Proceedings of the Sixth International Conference on Genetic Algorithms.* San Francisco, CA: Morgan Kaufmann.

Andre, David and Koza, John R. 1996. Parallel genetic programming: A scalable implementation using the transputer architecture. In Angeline, Peter J. and Kinnear, Kenneth E. Jr. (editors). 1996. *Advances in Genetic Programming* 2. Cambridge, MA: MIT Press.

Angeline, Peter J, and Pollack, Jordan B. 1992. The evolutionary induction of subroutines. In *The Fourteenth Annual Conference of the Cognitive Science Society.* Hillsdale, New Jersey: Lawrence Erlbaum Associates Inc., pp 236-241.

Baars, B. 1988. *A Cognitive Theory of Consciousness.* Cambridge: Cambridge University Press.

Beckers, R., Deneubourg, J. L., Goss, S., Pasteels, J. M. 1990. Collective decision making through food recruitment. *Ins. Soc.* 37, pp 259-267.

Beckers, R., Holland, O. E., Deneubourg, J. L. 1994. From local actions to global tasks: stigmergy and collective robotics. In *Artificial Life IV.* Brooks, Rodney A., and Maes, Pattie (editors). MIT Press.

Crosbie, Mark and Spafford, Eugene H. 1995. Defending a computer system using autonomous agents. In *Proceedings of the 18th National Information Systems Security Conference.* National Institute of Standards and Technology/National Computer Security Center.

Deneubourg, J. L., Goss S., Franks, N., Sendova-Franks, A., Detrain, C., and Cheretien, L. 1990. The dynamics of collective sorting: robot-like ants and ant-like robots. In *Simulation of Adaptive Behavior: from animals to animats.* Meyer, Jean-Arcady, and Wilson, Stewart W. (editors). MIT Press.

Dennett, Daniel C. 1991. *Consciousness Explained.* Boston: Little, Brown.

Fittkau, E. J., and H. Klinge. 1973. On biomass and trophic structure of the central Amazonian rain forest ecosystem. *Biotropica,* 5(1): 2-14.

Goldberg, David E. 1989. *Genetic Algorithms in Search, Optimization, and Machine Learning.* Reading, MA: Addison-Wesley.

Haynes, Thomas, Sen, Sandip, Schoenefeld, Dale, and Wainwright, Roger. 1995. Evolving a team. In *Proceedings of the AAAI-95 Fall Symposium Series on Genetic Programming.*

Holldobler, Bert, and Wilson, Edward O. 1990. *The Ants.* Cambridge, MA: The Belknap Press of Harvard University Press.

Kinnear, Kenneth E. Jr. (editor). 1994. *Advances in Genetic Programming.* Cambridge, MA: The MIT Press.

Koza, John R. 1992. *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* Cambridge, MA: The MIT Press.

Koza, John R. 1994. *Genetic Programming II: Automatic Discovery of Reusable Programs.* Cambridge, MA: The MIT Press.

Koza, John R. 1995. Gene duplication to enable genetic programming to concurrently evolve both the architecture and the work performing steps of a computer program. *Proceedings of the 1995 International Joint Conference in Artificial Intelligence.*

Koza, John R., and Andre, David. 1996. Evolution of iteration in genetic programming. In *Evolutionary Programming V: Proceedings of the Fifth Annual Conference on Evolutionary Programming.* Cambridge, MA: MIT Press. In Press.

Mataric, M. J. 1993. Designing emergent behavior: from local interactions to collective intelligence. In *From animals to animats 2.* Meyer, J. A., Roitblat, H. L., and Wilson, S. W. MIT-Bradford.

Minsky, Marvin. 1985. *The Society of Mind.* New York: Simon & Schuster.

Ohno, Susumu. 1970. *Evolution by Gene Duplication.* New York: Springer-Verlag.

Oster, G. F., and Wilson, E. O. 1978 *Caste and ecology in the social insects.* (Monographs in Population Biology, no. 12). Princeton University Press, Princeton, N.J.

Resnick, Mitchel. 1991. Animal simulations with *Logo: Massive parallelism for the masses. In Meyer, Jean-Arcady, and Wilson, Stewart W. (editors), *From Animals to Animtats: Proceedings of the First International Conference on Simulation of Adaptive Behavior.* MIT Press.

Ryan, Conor. 1995. GP robots and GP teams - competition, co-evolution and co-operation in genetic programming. In *Proceedings of the AAAI-95 Fall Symposium Series on Genetic Programming.*

Spector, Lee. 1996. Simultaneous evolution of programs and their control structures. In Angeline, P. and Kinnear, K. (editors) 1996. *Advances in Genetic Programming 2.* Cambridge, MA: MIT Press.

Cell Interactions as a Control Tool of Developmental Processes for Evolutionary Robotics

Peter Eggenberger
AI Lab, Department of Computer Science, University of Zurich
Winterthurerstrasse 190, 8057 Zurich, Switzerland
FAX: +41-1-363 00 35; Email: eggen@ifi.unizh.ch

Abstract

This paper describes new genetic and developmental principles for an artificial evolutionary system (AES) and reports the first simulation results. Emphasis is placed on those developmental processes which reduce the length of the genome to code for a given problem. We exemplify the usefulness of developmental processes with cell growth, cell differentiation and the creation of neural control structures which we used to control a real world autonomous agent. The importance of including developmental processes relies much on the fact that a neural network can be specified implicitly by using cell-to-cell communication.

1 Introduction

In the field of autonomous agents different approaches have been studied: One of them, the evolutionary approach, aims to produce increasingly sophisticated autonomous agents with no need to care about the details of the robots control structure. As others,(Nolfi et al.,1994; Cangelosi et al.,1994; Daellert & Beer,1994; Harvey et al.,1995), we are convinced that developmental processes have to be included in every artificial evolutionary system (AES). We suggest that biological concepts as developmental processes will show their worth when applied to evolutionary robotics. With the model proposed in this paper we can reduce the the length of the genome, because no explicit data about the connectivity pattern of the neural net are stored in the genome. The connectivity is mainly determined by the developmental processes and not directly encoded in the artificial genome itself. A good example that developmental processes can reduce the length of the genome and in this way the necessary information to determine complex structures is the animal coat pattern. Concretely, the patterning of a jaguar is not precisly genetically controlled, but emerges by a reaction-diffusion mechanism (Murray, 1993). The genome reflects the complexity of the neural network. Kolmogorov and Chaitin define the complexity of an object as a message determined by a minimal program coding this message (Chaitin,1975). This message is normally expressed in bits. Simple neural networks can be encoded by shorter genomes than more complex ones(e.g. a fully connected net needs only two active genes to be encoded). If the genome grows, then the information to encode the specified neural network has to grow too. The purpose of this work is to show that the introduced gene regulation mechanisms can control the main developmental processes and that the simulator is able to evolve functioning neural networks for autonomous agents.

2 Related Work

There are two main research groups within the evolutionary community. The first group puplishes papers about different methods how to encode genetically the structure of a neural network. (Wilson,1987) proposed a scheme which was based on a representation for biological development to study the evolution of multicellular systems. (Kitano, 1990) reduced the size of the genome using a graph generation grammar to encode neural network topologies. (Stork, Jackson and Walter, 1990) developed a system to evolve artificial networks. They introduced a structured genome which consists of two types of genes, control genes (also called enhancers) and structural genes. Enhancers control the expression of the structural genes, whereas the structural genes encode for fundamental aspects of the neural phenotype like cell adhesion molecules, neurotransmitters, receptors, etc. The activity of the enhancers is stored in an enhancer table and is genetically controlled. (Gruau and Whitley, 1993) encoded the developmental processes as grammar trees. In this approach the cells inherit their connections and no context sensitive development is possible. (Belew, 1990) also used a grammar to simulate developmental processes. His scheme is context sensitive, but it is restricted to pre specified neural network topologies (Fleischer and Barr, 1992) used a genetic coding to specify the developmental processes by means of differential equations. While from a biological perspective this approach is highly fruitful it is not suitable for autonomous

agents because of the high computational costs. (Sims, 1995) described a system for the evolution of artificial creatures that compete in a physically realistic simulation of a three-dimensional artificial world. Sims's approach has the problem that the shapes of the body parts and the neural control architecture of the creatures are genetically determined, i.e. there is no interaction with the environment during the developmental process.

The papers in the following group describe approaches to the evolution of control structures for real-world autonomous agents. (Harvey, 1992) developed the SAGA system to evolve neural networks. The problem in this approach is that the length of the genome grows quadratically with the network size (if all possible connections between the neurones are allowed), since the network structure has to be explicitly specified by the genome. (Vaario, 1994) has proposed a grammar-based simulation tool, in which the developmental process is described by a set of rules. A rule-based system bears the danger that certain properties of the system are defined by the designer rather than being emergent form the developmental process. (Cangelosi et al.,1994) presented a model of cell division and migration. (Michel and Biondi, 1995) introduced a biologically inspired model: They use morphogenetic mechanisms to evolve neural control structures for autonomous agents. As this model doesn't describe any mechanisms about cell differentiation it is not clear how different cells can result.

In section three we describe the used biological principles. A detailed description of our AES is given in section 4 and in section 5 we show a simple example of the used concepts. In the last section we discuss the results and describe our future research steps.

3 Biological Foundations of the AES

The following biological substances and mechansims are introduced in our artificial evolutionary system (AES).

3.1 Biochemical Substances

- Regulatory Units and Transcription Factors

  From genetics a lot is known about how genes are regulated. Good introductions and overviews about gene regulation are found in (Nicolis and Prigogine, 1977; Ptashne, 1992; Gilbert, 1994;).A biological cell contains substances, so called transcription factors which are able to link to specific places on the DNA molecule and activate or inhibit a single gene (in eukaryotes) or whole groups of genes (in bacterias). The specific loci where these transcription factors can bind are called promoters, enhancers or silencers depending on their binding place on the DNA and their function. It is important to note that products of genes can turn on or off other genes and that in this way the metabolism and different functions of the cells are controlled. The regulated genes are called structural genes.

- Cell Adhesion Molecules (CAM)

  Cell adhesion molecules (CAM) are proteins which determine the adhesion between cells and are of utmost importance for morphogenesis and the development of neural structures (Edelman,1989; Rutishauser and Jessell, 1988)). As different cells can have different CAM patterns on their surface so the connectivity between different cells vary. The appearance and disappearance of these important proteins in space and time will have a great impact on the number of cells, the form of the organisms or the connectivity of the neural networks.

- Cell Receptors

  For the cell-cell communication exist different receptors on the cells surface and for the regulation of the metabolism and the behavior of the cell. These receptors are in fact highly specialized proteins which are able to specifically recognize certain substances (e.g. hormones, neurotransmitters) from their environment. These receptors can vary in different cells which explains that the same signal can have a different impact on two different cells.

3.2 Cell differentiation

Cell differentiation is the basic mechanism to generate cellular diversity.Although all cells of an organism contain the same genome, they can become different because different subsets of genes are active. Cell differentiation is based on different mechansims from which we implemented two in our AES:

- cell lineage
- cell induction

During the process of cell lineage, the fertilized egg is divided into a multicellular structure and developmentally important substances are unevenly distributed to different cells and determines their fate. By this mechanism the cells are different from the beginning. Cell induction is a process during which one embryonic region interacts with another to influence its differentiation. As cell induction depends either on cell-contact or diffusible substances, it is position-dependent and the specificity lies in the interaction of the diffusable substance with the receptor which is specific to a particular substance.

3.3 Development

The main developmental stages of an organism like fertilisation, cell cleavage, gastrulation, organogenesis and maturity are controlled by the following main processes

(Edelman, 1989; Gilbert, 1994): cell division, cell differentiation, cell adhesion, cell death and cell migration. Out of them we introduced the first three in our model whereas cell death and cell migration are not yet included for computational reasons and not because we consider them as unimportant. All five mechanisms are controlled by genetic factors as well as epigenetic mechanisms and are dynamically adjusted in space and time.

The development of the pattern of synaptic connections in the nervous system can be divided in five major steps (Kandel et al;1995):

1. the precursor cells divide and diversify, giving rise to glial cells and immature neurons

2. by migration the immature cells leave the germinal zone and find their final places

3. neurons extend their axons into the vicinity of their future targets

4. the neurons and selected target cells are connected by synapsis

5. the initial pattern of synaptic connections is pruned by competition among neurons for trophic factors and by physiological activity to form the mature pattern of connections.

4 Description of the Concepts Implemented in the Artificial Evolutionary System (AES)

The following subsections describe the different implemented concepts of our AES which are under the control of a genetic algorithm.

4.1 Genome

The artifical genome is implemented as a string of integer, but also a structure is introduced to control the activity of the genes. We divide the genome in parts with the numbers 0, 5 and 6. (See the first line of figure 1). You can see that every regulatory unit is marked at its end with a five. The structural genes can have either a zero at its end or a six. The zero mark only the end of a gene where as a six has the additional information that the range of influence of the regulatory unit stops here. After a six a new regulatory unit is inserted.

At the beginning all the genomes in the population have this sterotyped structure. The number of regulatory units and the amount of genes which they regulate are predefined constants. In this structure the algorithm inserts randomly the numbers 1,2,3 and 4. In figure 1 you can see an example of such a structure in where a regulatory unit controls the activity of two structural genes. In fact, a regulatory genes is nothing but a switch which activates or desactivate the associated structural genes. The detailed structure of a gene is illustrated in figure 2).

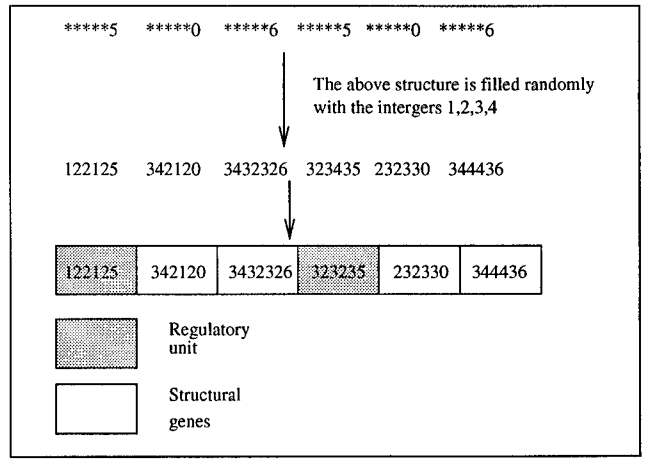


Figure 1: The artifical genome is implemented as a string of numbers.The genome is separated in regulatory units and structural genes. Regulatory units are used to activate or inhibit the activity of the structural genes which are on the right side to the regulatory unit. The range of control for a regulatory unit ends at the next regulatory unit.

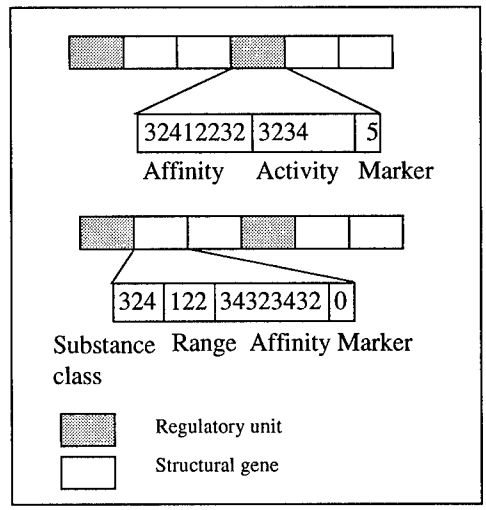


Figure 2: Gene structure. The code of a regulatory unit has three parts. How these parts are used is explained in the text. A structural gene has four different parts. The first three integers are used to assign to the gene a substance class. The second part encodes a range which is used to calculate either a diffusion constant of a transcription factor or the range to how many neighbors a neuron will search for a contact (cell adhesion molecule) depending on the substance class. The third part is used to define an affinity function and the last integer is used to separate the genes and to distinguish between regulatory units and structural genes

4.2 Classes of Gene products

The activation of a structural gene means that an artificial substance is produced out of four different classes. The following artificial substance classes are defined:

- transcription factors. This class is used to regulate the activity of genes.
- cell adhesion molecules. They are used to build connections between neurons.
- receptors. Receptors are used to regulate the communication between the cells.
- artificial functions. This class is used to define to determine wheather a cell should divide or not.

The first three integers of the code of a structural gene determine to which substance class a gene will belong. The artificial substances are represented directly by the sequence of intergers of the gene code.

4.3 Regulation of the Gene Activity

If a cell contains a transcription factor, its code is compared with the code of all regulatory sites in this cell. Depending on a defined affinity function the regulatory site is then activated or inhibited. Given the length of

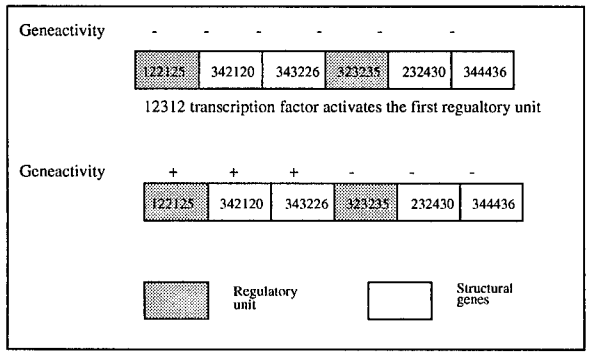


Figure 3: Gene regulation. A transcription factor is compared with the code of a regulatory unit and if the affinity is high enough the regulatory unit will be activited. This will have the effect that the structural genes of that regulatory unit also will be activated. The principle of inactivating a regulatory unit is analogue.

a regulatory site and its transcription factor the affinity function determines the affinity in comparing the code sequences. To decide if a regulatory unit is activated or inhibited we have implemented the following procedure: the number of equal integers in both codes are counted (homologous pairs) as well as the number of the following pairs (1,4),(4,1), (2,3) and (3,2) (anti-homologuous) are counted. We substract the two numbers and activate the regualtory unit if the number exceeds a predefined positive threshold number or inactivate it if the number is smaller than the inactivation threshold. As several substances can compete we calculate all affinities of all substances, the majority of all influences then will determine the activity of the particular gene or group of genes. In other words, there is a competition between the transcription factors for their regulatory site. If a regualtory site is activated also its structural genes will be activated as can be seen in figure 3.

Example: The following two strings contain two homologuoes pairs and four anti-homologous pairs.

12343212 code for the regulatory unit

42312334 code for the transcription factor

If the threshhold would be two and no other transcription factor influences the regulatory unit, it would be activated.

4.4 Cell Differentiation

We say that two cells are different, if they contain different subsets of active genes in the genome. To get different cells we implemented two different mechanisms: cell lineage and cell induction.

4.4.1 Cell lineage

We used the activity part of the regulatory unit (See figure 2) to decide which regulatory sites of the genome in the first cells are active or not. The first cell will read the first integer in the activity part of each regulatory unit and will activitate this unit if the gene is even and inactivate it if the integer is uneven. The second cell will do the same with the second integer of the activity part and so on. We restricted the system to four numbers which has the consequence that maximally four different cells will emerge at the beginning.

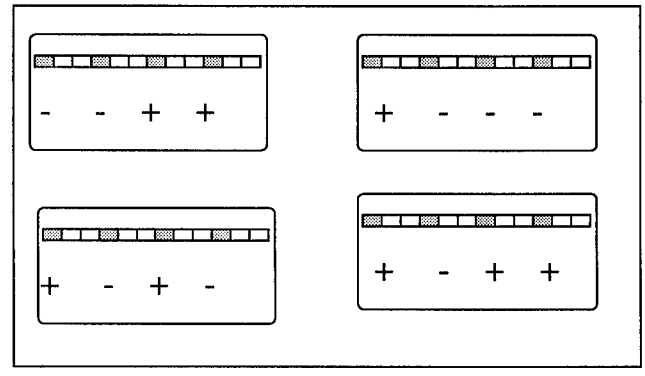


Figure 4: Cell Lineage: In the genome at a predefined location we use the activity part of a regulatory unit to decide which regulatory unit in the first cells are on or off. The result will be, that from the beginning different cells are created.

4.4.2 Cell induction

Cell induction is a fundamental mechanism of cell differentiation. To simulate cell induction we implemented

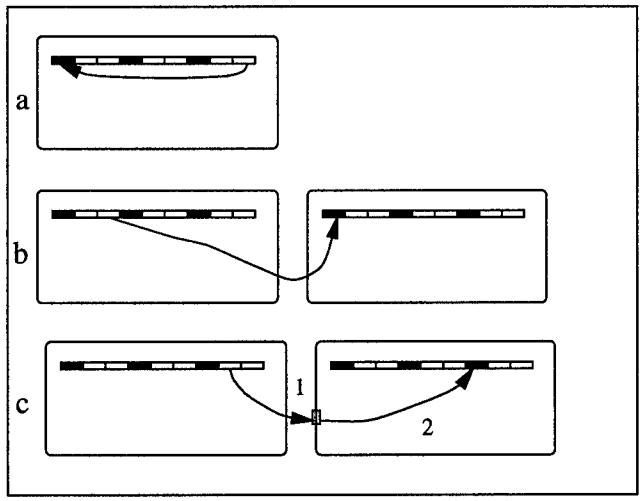


Figure 5: Intercellular gene regulation by activators and repressors.The following regulation scheme is implemented: a. intracellular regulation b. intercellular regulation c. The intercellular communication depends on a receptor (1), which sends an activating signal to a regulatory site (2), if the affinity between the receptor and the transcription factor is high enough.

three different possible pathways to exchange information between cells (Figure 5). First, there are substances which do not leave the cell and which regulate the activity of genes. Second, there are substances which can penetrate the cell wall and activate all cells which are near by. Third, there are specific receptors on the cell surface which can be stimulated by substances. If a transcription factor has a high enough affinity to the receptor, a gene or a group of genes is turned on or off. Only those cells which have a specific receptor on the cell surface will respond to a certain substance (filtering of the communication between cells).

4.5 Construction of a Cell

After the process of cell differentiation is finished, the different active genes will determine which substances are produced in a cell. These substances are stored in lists for further use.

At this point of the development the artificial cells have a structured genome, a list which holds the activity of the genes (which can change dynamically) and different lists which represents different substance classes that are contained in the cell. These lists are used differently depending which substance classes they represent.

4.6 Cell Division

To increase the number of the cells we used two different schemes. In the first case, we have two parameters to fix the number of the cells on a grid.

In the second case we wanted to see if the proposed model is also able to simulate cell growth. At the beginning of a simulation one cell is put on a 2-dimensional grid.

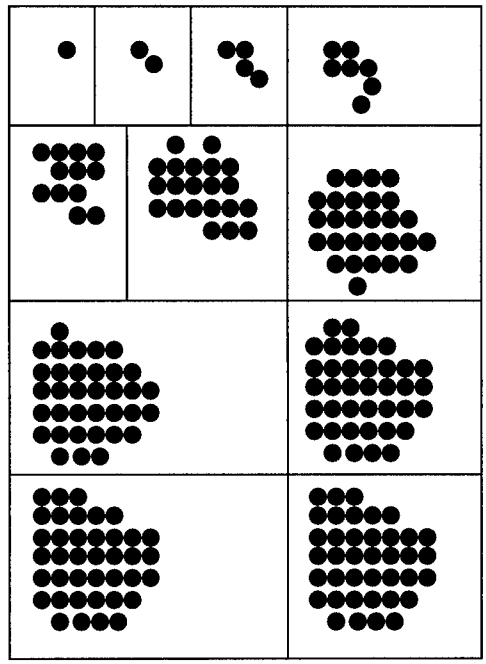


Figure 6: Example of the growth of a cell cluster. The cells contain two regulatory unist and two structural genes for cell division and the production of a transcription factor which is able to turn of the cell division gene if the concentration reaches a certain level. After ten cell divisions the growth stops.

If the structural gene for cell division in a cell is active, the cell divides. During cell division one cell rests at the same place on the grid, whereas the other is placed on one of the neighboring places of the grid. In case that all places around a cell are already occupied, the cell is not able to divide, although the gene for cell division is on. To be able to simulate cell growth, an additional feature had to be implemented. As the cells are on a grid, at every grid position a list with all transcription factors is stored. These transcription factors stem from other cells on the grid and can vary in type and concentration. The gene activity of a gene is now not only dependent on the usual affinity function, but also on the concentration. In the case, there the affinity of a transcription factor is not able to turn off the gene for cell division the number of cells will increase. But the effect of increasing the cell number will increase also the concentration of the diffusable transcription factor. At a certain moment, due

to its increased concentration, the transcription factor will turn off the gene for cell division and the growth of the cell cluster will stop.

4.7 The Neurons and their Connections

4.7.1 Description of the Used Neurons

We use standard artificial neurons which obey the following equations:

$n_i(t+1) = \sigma(\sum_{i=1}^{n} \omega_{i,j} n_j(t) - \theta)$.

The σ-Function is defined as

$\sigma(x) = \frac{1}{1+exp^{-x}}$

The output of the neuron i is represented as n_i. Time t is taken as discrete. The synaptic weight $\omega_{i,j}$ represents the strength of the connection between neuron j and neuron i. During a time step the neuron sums up all active synapses and if a fixed threshold θ is exceeded the output of the neuron will fire.

4.7.2 Connections between two Neurons

As cells can become different, they will express also different substances. To connect two cells or neurons, there are two different types of adhesion molecules. These are stored in lists in the cell and are used differently. The members of the first list of one cell are compared to the second list of another neuron. If two adhesion molecules of the two different lists have a high affinity to each other (again the integers of the two substances are compared), a link from the first cell to the second cell is established. Note, that the direction of the link is given by the two types of lists. Biologically, we can imagine that the first

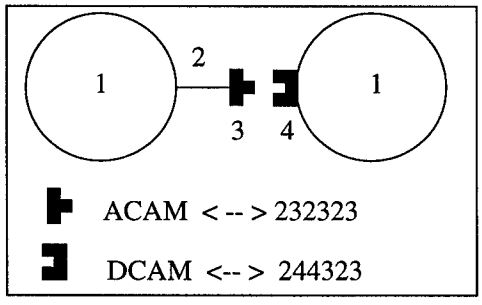


Figure 7: The two different cells have two lists which we call A-CAM and D-CAM list. (A for axon and D for dendrite). The two lists are compared and if two members of the two lists have an affinity which is high enough, a link between the cells is established. Note the the CAMs are represented as code strings which encode the affinity between the two sorts of CAM's

list are adhesion molecules on the axon, and the members of the second list are adhesion molecules on the dendrites. If two or more links to the same cell are possible, the substance with the greatest affinity is chosen. For each cell we maintain a list of cells to which it is connected. For the first tests we used only four different weight values. To make the choice between the weights we counted the integers used (1,2,3,4) and depending on the counted number of each used integer we assign one of the four weights to the specific synapse. But not all cells will look through all the lists of all cells and will decide to whom a link should be established. The code of the CAM is used to define a search range. This range determines the maximal distance of two cells in the grid which can connect with each other. As explained in figure 2 the gene has a part named range, which is used to determine how long the search range for a neuron is. Minimally, the cell looks only at the nearest neighbors and maximally, a cell tries with every cell to built a contact.

4.8 Establishing a Link to the Sensors and Motors of the Real World Robot

The newly developed neural network has to be linked to the sensors and motors of a real world robot. How can we link the evolved neural network to the fixed number of sensors and motors of the robot? For this we had to define sensory and motor cells which hold a list with adhesion molecules (which are arbitrarily defined by hand) and to which the other cells can connect. In this way the algorithm is able to connect the developed neural network to the motors as well as to the sensors.

5 Advantages of an Indirect Encoding for Synapses

An encoding scheme which separates the genome in regions which contain information about a single cell (Harvey,1992; (Nolfi et al.,1994) has the disadvantage that the genome will grow with an increasing number of cells. This makes it difficult to find a solution for a given problem, if the number of cells grows. In contrast to such

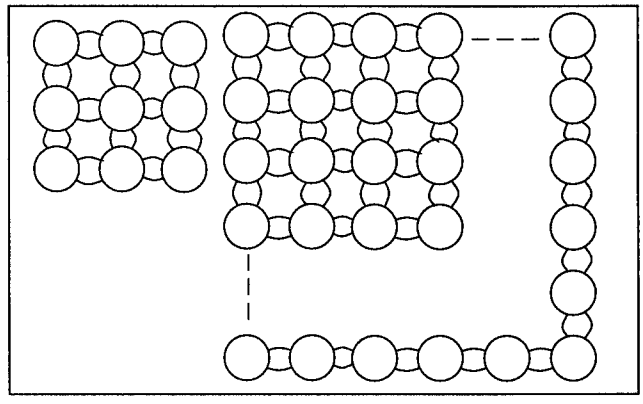


Figure 8: The little homogeneous net at the left has the same genome length as the large one.

direct encoding schemes, in our approach the genome

grows not necessarily with the number of cells. To show this we give an example for a simple network. Suppose that the task is to evolve a neural network structure in which every cell is connected to its nearest neighbors. Say, that at the beginning there are nine cells. For this task the genome needs one regulatory unit with two structural genes which encode for one CAM for the axon and one for the dendrites. The affinity of these two CAM's should allow a link and the regulatory unit has to be active. With this simple genome we are able to link the nine cells to each other (Figure 8). If we now enlarge the number of neurons in the network to a much bigger number, the genome would also link neighboring cells. This Gedankenexperiment is one of the main reasons why we think it important to use developmental processes to evolve control structures for robots. The genome should only grow, if the information to encode the neural connectivity also grows. Simple or homogenous neural networks should need only short genomes.

6 Results

At the basis of our AES is a genetic algorithm. The

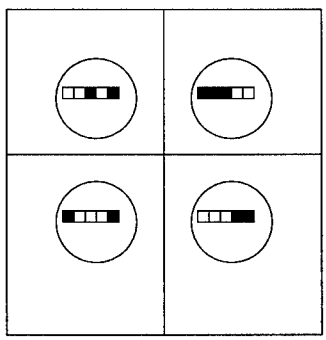


Figure 9: At the beginning four different cells are produced. Only regulatory units are shown in the cells. Black boxes indicate an active regualtory unit, a white one an inactive one.

tasks are in the first example object avoidance with the goal to examplify the advantage of an approach with developmental processes. In the second task we evolved a neural controller which moves, avoids objects and approaches a light source.For this two simulations we used a predefined number of cells.

6.1 Evolving Neural Control Structures for an Autonomous Agent

The goal of this section is to give an example of the interactions of the different concepts which are described in section 4. At the beginning 200 genomes are prepared. All genes in the genomes of each cell have the same length and the number of regulatory units and structural genes is also the same. In the first step the routine "Cell induction" is called and the first four cells have different subsets of active genes (See figure 9). In this simulation

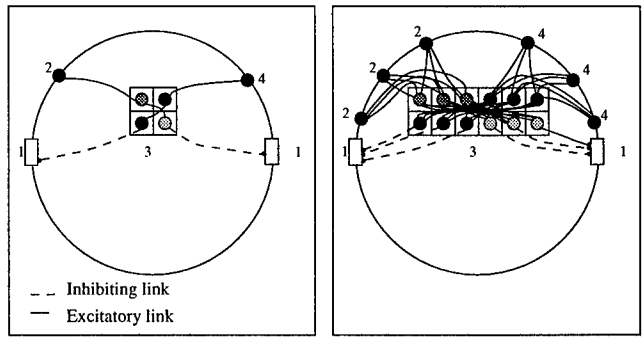


Figure 10: The image shows two images of an autonomous agent (Khepera). Two Braitenberg models for object avoidance are evolved assuming that the robot has already an initial speed. The image at the left shows the found solutions. In the image at the right we see a more complicated solution for the same task. The importance of this example is that the genome of the two solutions is the same. 1. Motor 2. Infrared sensors which is also represented as a cell with a predefined A-CAM. 3. Neural network 4. Infrared sensor which is also represented as in 2, but which has another A-CAM.

we leave the number of cells fixed, because we wanted to show that the genome length can be the same even if the number of cells is multiplied. The neurons are developed and depending on the set of active genes the neurons can have different substances belonging to different classes. After the neural network had developed,

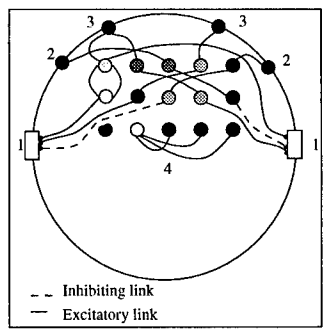


Figure 11: An example of an evolved neural network for obstacle avoidance and light approach. 1. Motor cells 2. Infrared sensors 3. Ambient light sensors 4. Neural Network.

we tested it first on a simulator if it was worth to test it on a real world robot (Khepera). For the first task we increased the fitness,

- if the robot sees an object but avoids a collision

and for the second task, we increased the fitness,

- if the robot moves away from its inital position
- if the robot is near the light source
- if the robot sees an object but avoids a collision

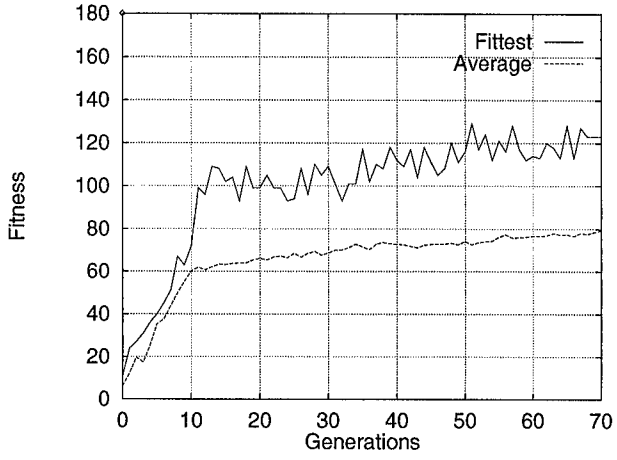


Figure 12: The average fittness and the fitness of the best individual is plotted for every generation.

If one of the simulated robots had reached the best fitness the program was used to control a Khepera and we tested the behavior. As we have not yet an automatic data assignement for a real world robot this was a time consuming task although the behavior of the simulated robot with the real one was quite similar. As genetic operators we used only mutation and cross-over. In the average the system needed 10 to 20 generations to find a solution for the first task and 60 to 80 generations for the second task (See Figure 12). The genome of the best individual is unchanged to guarantee that the algorithm converges (Rudolph,1994).

7 Discussion

In this work we argued that biological ideas are usefull and applicable to the field of evolutionary robotics. Using important developmental processes we showed that cell growth, cell differentiation and the development of useful neural control structures are embodied in our method. Especially important is the fact that the genome will not necessarily grow, if the number of neurons or cells will grow.

Presently, we are working on three main lines:

1. Analysis of the simulator to explore its limits. We will analyse statistically our AES and test the different results with different initialisations of the random generator and the different possibilities of the genetic operators which can be introduced if one uses structured genomes. Interchanging and duplicating genetic material seems very promising to us, especially as these operators change possible interactions between cells.

2. Extensions of the used methods for more cells. If the number of cells will increase, the introduced local methods need an extension. To build connections between distant groups of cells, additional methods should be introduced.

3. The evolutionary creation of the robots' morphology. As there is an interplay between the neural control structure and the sensory and motor systems on a robot, we want to investigate the interactions of the robots form and its behavior.

8 Acknowledgements

Many thanks to Daniel C. Meier, Rolf Pfeifer, Ralf Salomon and René Schaad for helpful discussions.

References

Belew, R. K. (1990). Evolution, learning, and culture: computational metaphors for adaptive algorithms. *Complex Systems*, **4**, 11–49.

Cangelosi, A., D. Parisi, and S. Nolfi (1994). Cell division and migration in a 'genotype' for neural networks. *Network*, **5**, 497–515.

Chaitin, C. (1975). Randomness and mathematical proof. *Scientific Americain*, **232**, 47–58.

D. G. Stork, B. Jackson, S. Walter (1992). Non-optimality via pre-adaptation in simple neural systems. In *Paper presented at the Artifical Life II,Santa Fe, Mexico.*

Daellert, F. and R. D. Beer (1994). Toward an evolvable model of development for autonomous agent synthesis. *Artifical Life IV*, 246–257.

Edelman, G. M. (1989). *Topobiology:An Introduction to Molecular Embryology*. New York:Basic Books.

Fleischer, K. and A. Barr (1992). A simulation testbed for the study of multicellular development: The multiple mechanisms of morphogenesis. In (Langton 1994), pp. 389–416.

Gilbert, S. F. (1994). *Developmental Biology*. Massachusetts: Sinauer Associates, Inc.

Gruau, F. and D. Whitley (1993). The cellular developmental of neural networks: the interaction of learning and evolution. Technical Report 93-04, Laboratoire de l'Informatique du Parallélisme, Ecole Normale Supérieure de Lyon, France.

Harvey, I. (1992). Species adaptation genetic algorithms: The basis for a contiuniung saga. In F. J. Varela and P. Bourgine (Eds.), *Toward a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life*, pp. 346–354. MIT Press/Bradford Books, Cambridge, MA.

Harvey, I., P. Husbands, and D. Cliff (1995). Evolutionary approaches. In *Toward the Practice and Fu-*

ture of Autonomous agents, Monte Verita, Ticino, Switzerland.

Kandel, E. R., J. H. Schwartz, and T. M. Jessell (1995). *Essentials of Neural Science and Behavior.* Appleton & Lange.

Kitano, H. (1990). Designing neural networks using genetic algorithms with graph generation system. *Complex Systems*, **4**, 461–476.

Langton, Christopher G. (Ed.) (1994). *Artificial Life III: Proceedings of the Workshop on Artificial Life*, Reading, MA. Addison-Wesley. Workshop held June, 1992 in Santa Fe, New Mexico.

Michel, O. and J. Bondi (1995). From the chromosome to the neural network. In *Proceedings of the interantioal conference on artificial Neural Networks and Genetic Algorithms (ICANNA'95).*

Murray, J.D. (1993). *Mathematical Biology* (Second,Corrected Edition ed.). Biomathematics Texts. Springer-Verlag.

Nicolis, G. and I. Prigogine (1977). *Self-Organization in Non-equilibrium Systems.* New York:Wiley.

Nolfi, S., D. Floreano, O. Miglino, and F. Mondada (1994). How to evolve autonomous robots: Different approaches in evolutionary robotics. In P. Meas R. A. Brooks (Ed.), *Artificial Life IV.* Cambridge, MA: MIT Press.

Ptashne, M. (1992). *A Genetic Switch. Phage lambda and Higher Organisms.* Cell Press and Blackwell Scientific Publications.

Rudolph, Günter (1994). Convergence analysis of canonical genetic algorithms. *IEEE Trans. Neural Networks, Special Issue on Evolutionary Computation*, **5**, 96–101.

Rutishauser, U. and T.M. Jessell (1988). Cell adhesion molecules in vertebrate neural development. *Physiological Reviews*, **68**, 819–857.

Sims, K. (1995). Evolving 3d morphology and behavior by competition. *Artificial Life*, **1**, 353–372.

Vaario, J. (1994). Modelling adaptive self-organization. In *Proceedings of the Fourth International Workshop on the Synthesis of Living Systems.*

Wilson, S. W. (1987). The genetic algorithm and simulated evolution. In C. Langton (Ed.), *Artificial Life. A Proceedings Volume in the Santa Fe Institute Studies in the Sciences of Complexity*, pp. 157–165.

Evolution of the Sensorimotor Control in an Autonomous Agent

Susanne A. Huber, Hanspeter A. Mallot, Heinrich H. Bülthoff
Max-Planck-Institut für biologische Kybernetik
Spemannstr. 38, 72076 Tübingen, Germany.
phone: ++49 (0)7071 - 601 - 608/603/600
fax: ++49 (0)7071 - 601 - 616
email: huber/ham/hhb@mpik-tueb.mpg.de

Abstract

Visually guided agents are introduced, that evolve their sensor orientations and sensorimotor coupling in a simulated evolution. The work builds on neurobiological results from various aspects of insect navigation and the architecture of the "Vehicles" of Braitenberg (1984). Flies have specialized visuomotor programs for tasks like compensating for deviations from the course, tracking, and landing, which involve the analysis of visual motion information. We use genetic algorithms to evolve the obstacle avoidance behavior. The sensor orientations and the transmission weights between sensor input and motor output evolve with the sensors and motors acting in a closed loop of perception and action. The influence of the crossover and mutation probabilities on the outcome of the simulations, specifically the maximum fitness and the convergence of the population are tested.

1 Introduction

In this work autonomous agents are introduced which navigate through a virtual world. Genetic algorithms are applied to evolve their visually guided control mechanisms and generate a sensorimotor coupling which enables them to survive in the environment. In particular, the behavioral module for obstacle avoidance is studied. For the task a visuomotor program is generated with the sensors and effectors acting in a closed loop of perception and action, thus effecting a permanent sensorimotor interaction.

In information processing the architecture of autonomous systems is decomposed into a chain of functional modules such as perception, information processing in a central unit and the execution and output of information. In other approaches the architecture is decomposed into task-achieving modules, which, in combination, produce the complex, "emergent" behavior of biological (Tinbergen, 1953) and artificial systems (Brooks, 1986; Flynn & Brooks, 1989). Starting from the assumption that perception and action – sensor input and motor control – did not develop independently from each other, but are a coupled system – they have to be investigated in a closed loop. Braitenberg demonstrated with his "Vehicles" that even with simple architectures, it is possible to conceive of autonomous agents that can exhibit complex emergent behavior.

By studying the behavior of insects and the underlying neural mechanisms (for review see Egelhaaf & Borst, 1993), the architecture of biological navigation systems has been investigated. For our agent, the most important biological insight is that insects navigate mostly by evaluating visual motion information by means of neurons tuned to specific motion patterns (matched filters). The spatial localization of the receptive fields of these neurons is optimized with respect to certain behavioral tasks.

Franceschini and his colleagues demonstrate that the principle of motion vision can be used for navigational tasks in simulated and real agents (Franceschini, Pichon & Blanes, 1992). Cliff, Husbands and Harvey (1994) show the efficacy of using genetic algorithms to evolve concurrently the visual morphology along with the control networks. Here, we attempt to combine these approaches by evolving a competence for obstacle avoidance through simultaneous adaptation of sensor parameters and the sensorimotor coupling.

In the section 2, results from the research on the visual system of flies are reviewed and in section 3 the architectures of two types of autonomous agents are described. In section 4 the genetic algorithms used here are introduced, followed by the results of the simulations.

2 Perception of motion and visuomotor control in flies

The resolution of the compound eyes of flies is much coarser than that of human eyes and thus the perception of shape is more difficult. Hence, for visual orientation the detection of motion plays a more prominent role. While the insect is navigating through a stationary environment the images on the retinae are con-

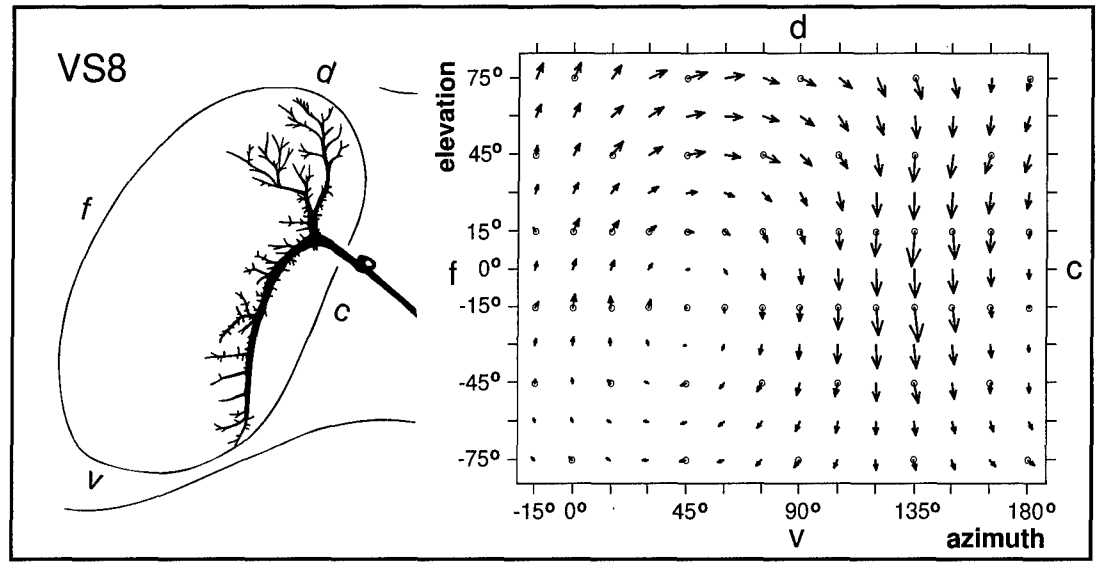


Figure 1: *Tangential cell VS8, responding maximal to rotations around the axis with azimuth* 45° *and elevation* −9°. *The measured flow vectors are indicated by a circle, the vectors in between result from interpolation (Hengstenberg et al. (in press)).*

tinuously changing. This image flow depends on both the trajectory through the world and the structure of the environment. In a stationary environment the rotational flow field contains information about the egomotion only, whereas translational fields contain both information about the structure of the environment and information about the movement of the observer. Objects nearby cause a larger image flow than objects further away (Longuet-Higgins & Prazdny, 1980).

As a model for motion perception in insects, Reichardt & Hassenstein proposed a correlation detector (Hassenstein & Reichardt, 1956; Reichardt, 1961) which correlates temporal modulation of image intensities in two neighboring ommatidia. Here we use a version of the correlation-type movement detector, where the visual signals first are temporally highpass filtered, making the motion detector independent of background illumination. From the investigation of the behavior of flies under controlled experimental conditions (Götz, 1964; Reichardt, 1979) as well as with freely flying flies (e. g. Wehrhahn, Poggio, Bülthoff, 1982; Wagner, 1985), different visuomotor subsystems for course control and object fixation have been described. This behavior corresponds to anatomical structures that are located in the visual area of the fly's brain. The so-called tangential cells (Hausen, 1982) (in the lobular plate – a section of the visual system) are known to play a prominent role in the detection of egomotion and thus are essential for the visuomotor control that compensates for deviations from the intended course. One example is shown in Fig. 1 (Hengstenberg, Krapp, Hengstenberg (in press)). These cells have large receptive fields, which together cover most of the visual field.

A second system detects image expansion signalling approaching objects or obstacles in the heading direction. Motion in small parts of the visual field which result from movements of small objects are detected by a third class of cells; it is used for the tracking of other flies (Egelhaaf & Borst, 1993). Position and extension of the receptive fields of these neurons in the visual system of the fly and also the specialization to certain motion patterns are essential for the course control of the fly.

3 The autonomous agents

In order to build an artificial agent that navigates using strategies as they are known in flies (see section 2), the idea is to evolve the matched filters and the sensorimotor coupling for certain behavioral tasks. In flies, each filter consists of a field of motion detectors with specialized orientations (Fig. 1). In this work we start with an agent that has only four visual sensors and two motors. Two sensors form a movement detector and the outputs of the two detectors are coupled via transmission weights to the two motors. The autonomous agent gathers information about its egomotion and the environment by evaluating the motion signals from the detectors. The orientations of the visual sensors determine which part of the motion field is used to navigate through the unknown environment.

They are thus a particularly simple case of matched filters for the course control. Genetic algorithms are used to evolve the sensor orientations and the coupling weights

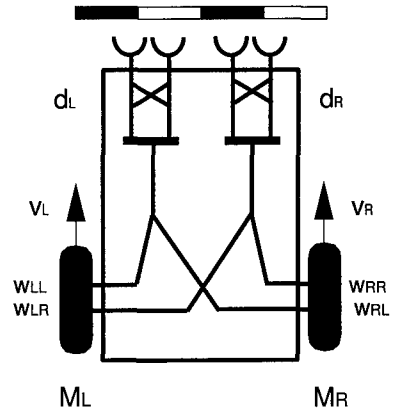


Figure 2: *The agent has four sensors, the orientations of which is given by the angles azimuth Φ and inclination Θ. The outputs of these detectors d_L and d_R are connected to the motors M_L and M_R via the sensorimotor coupling.*

between sensors and motors for different types of behaviors.

The input to each sensor is computed by "ray tracing" (Foley et al., 1987) where the intensities of single points – at the intersection of the line of sight with the visible surfaces – are averaged over a given number of sampling points. The orientations of the optical axes of the two sensors on one hemisphere of the visual field are evolved by the genetic algorithm. The other pair of sensors is positioned bilaterally symmetric on the other hemisphere. The time constants of the lowpass filters of the correlation motion detector are fixed ($\tau_{LP1} = 2.0\text{s}, \tau_{LP2} = 5.0\text{s}$). The matrix

$$\mathbf{W} = \begin{pmatrix} w_{LL} & w_{LR} \\ w_{RL} & w_{RR} \end{pmatrix} \tag{1}$$

contains the transmission weights for the sensorimotor coupling of the outputs d_L and d_R of the two motion-detectors with the two motors M_L and M_R. The velocity of the system is proportional to the force of the two motors, each motor producing a basic velocity $\mathbf{v}_0$ which is modulated by the visual information:

$$\begin{pmatrix} v_L \\ v_R \end{pmatrix} = \mathbf{v}_0 - \mathbf{W} \begin{pmatrix} d_L \\ d_R \end{pmatrix} . \tag{2}$$

The velocity V in the heading direction and the anglular velocity are:

$$V = \frac{1}{2}(v_L + v_R) \quad \text{and} \quad \dot{\varphi} = \frac{v_L - v_R}{c} . \tag{3}$$

where $c = 10\text{cm}$ is the distance between the wheels. The system has two degrees of freedom: rotation around the vertical axis and translation in the heading direction. In the simulations the numerical accuracy is set to 10^{-6} simulating a small amount of noise.

3.1 Agent of type 1:

Here the angular aperture of each sensor is 10° azimuth × 10° elevation. We average the intensity of 10 × 10 sampling points to compute the visual input to each sensor. The basic velocity of the two motors is constant at $v_0 = 10\text{cm/s}$.

The agent moves through a tunnel which has a sinusoidal pattern ($\lambda = 1\text{m}$) mapped onto the walls, the floor and the ceiling. The width and height of the tunnel are 6m, the length is 100m. The elevation of the agent in the tunnel is kept constant at 3m. During evolution the system has to avoid two walls in the tunnel and maintain a safe distance of 15cm while navigating around the obstacles. The two walls are at $x = 15.0\text{m}$, $0.0\text{m} \leq y \leq 3.0\text{m}$ and $x = 35.0\text{m}$, $-3.0\text{m} \leq y \leq 0.0\text{m}$.

3.2 Agent of type 2:

For the agent of type 2 bilateral symmetry is assumed for the orientation of the motion detectors – as for agent of type 1 – and in addition for the transmission weights from the detector outputs to the motors. The angular aperture – being the same for all four sensors – is evolved. In order to keep the simulation time small, horizontal line sensors are used. The number of sampling points varies with the angular aperture of the sensors, the sampling base is kept constant at 1°. In addition the constant basic velocity v_0 of the two motors is a parameter optimized during evolution.

We run two blocks of simulations: in block 1 a sinusoidal pattern with the wavelength $\lambda = 2\text{m}$ is mapped onto the walls, ceiling and floor. Here the tunnel is 110m long and closed by a wall at both ends, the width and height is 4m. Four additional walls are placed at $x = 9\text{m}, 50\text{m}; 0.0\text{m} \leq y \leq 2.0\text{m}$ and $x = 19\text{m}, 80\text{m}; -2.0\text{m} \leq y \leq 0.0\text{m}$. The agent maintains a constant height of 2m.

In block 2 a random-dot pattern is used and walls are placed at $x = 9\text{m}, 50\text{m}; 0.0\text{m} \leq y \leq 2.0\text{m}$ and $x = 25\text{m}, 80\text{m}$; $-2.0\text{m} \leq y \leq 0.0\text{m}$. The tunnel is open at the end. The agents have to maintain a save distance of 10 cm from the walls.

4 Simulated evolution

4.1 The genetic algorithm

In a simulated evolution – using genetic algorithms – the autonomous agents adapt to the environment by generating an obstacle avoidance behavior. The orientation of two sensors and the transmission weights for the sensorimotor coupling are evolved. These parameters are encoded as a Gray-coded bitstring. Starting with a random initial population of bitstrings, each new generation is obtained by the following procedure:

1. Raw fitnesses are scaled linearly such that average fitness $\bar{f}$ is unchanged and maximal fitness is scaled to $n\bar{f}$ for some constant $n \geq 1$ (Goldberg, 1989). The coefficient n is set to:

$$n = \min\{n_c, n_0\} \tag{4}$$

where n_c is a constant value and

$$n_0 = \frac{f_{\max} - f_{\min}}{\bar{f} - f_{\min}}. \tag{5}$$

For the case $n_c > n_0$, scaling causes negative fitness values if n_c is used. Therefore n_0 is applied instead. Here the scaling still leaves the average fitness $\bar{f}$ unchanged but leads to a scaled $f_{\min} = 0.0$ – preventing negative fitness values – and a scaled $f_{\max} = n_0\bar{f}$.

2. The number of offspring of each individual, N_i, is obtained by a random procedure such that the expectation of N_i is proportional to the scaled fitness ("roulette-wheel" selection). In terms of the raw fitness, we have

$$E(i) = \frac{N-1}{N}(1 + (n-1)\frac{f_i - \bar{f}}{f_{\max} - \bar{f}}), \tag{6}$$

where N is the total population size. The factor $(N-1)/N$ is needed since only $(N-1)$ individuals of the new generation are obtained by this scheme.

3. The selected parents exchange their genetic material by one-point crossover. In addition point-mutation is used to introduce new genetic material into the population.

4. The individual with maximal fitness is transferred to the next generation automatically ("elitist-strategy"; Davis, 1991).

	$C_{++}M_{+}$	$C_{++}M_{0}$	$C_{0}M_{+}$	$C_{0}M_{++}$
p_c	0.7	0.7	–	–
p_m	0.01	–	0.01	0.1

Table 1: *Probabilities for mutation p_m and crossover p_c tuned by hand.*

4.2 Probabilities for mutation and crossover

The optimal parameter settings for the mutation and crossover probabilities are not yet fully understood. The genetic algorithms of Holland (1975) use crossover as the primary operator with mutation being of secondary importance. In general, mutation has a high exploratory power independent of the diversity of the population (Spears, 1993). The power of crossover lies in the construction and preservation of individuals of high fitness. The exploratory power of crossover is limited as the population loses diversity and the individuals become more and more similar. According to Spears (1993) the choice of the genetic operators depends on whether the whole population should gain a high fitness – here using crossover is of advantage – or one optimal individual is to be found, in which case use of mutation is sufficient to obtain comparable and better results. In order to test the optimization behavior of the genetic algorithm for different crossover and mutation probabilities (p_c and p_m), we selected four conditions (see Table 1) in our first block of simulations.

4.3 GA 1

For the agent of type 1 the angles azimuth Φ and inclination Θ – describing the sensor orientations – are encoded with 4 bits each, in the range from 5° to 175° with a stepwidth of 11.3°. The weights of the sensorimotor coupling can take the eight real values $[\pm 0.5, \pm 0.1, \pm 0.05, \pm 0.01]$; they are encoded with 3 bits each. The length of the resulting bitstring is thus $4 \times 4\text{bits} + 4 \times 3\text{bits} = 28\text{bits}$. The raw fitness f is set to zero if the agent bumps into a wall within 400 time steps. Individuals which survive the 400 time steps in the tunnel without colliding with the walls, receive the fitness:

$$f = \frac{x(t = 400)}{\dot{\varphi}_{\max}}. \tag{7}$$

Here x is the component of the position on the center axis. Additionally, in order to keep the change in rotations in a limited range, the maximal angular velocity generated on the path $\dot{\varphi}_{\max}$ is used. A population size of 100 and $n_c = 1.2$ are used.

4.4 GA 2

The angles Φ and Θ of the agents of type 2 are again encoded with 4 bits each. The angular aperture is encoded with 3 bits in the range of 10.0° to 27.5° with a stepwidth of 2.5°. The transmission weights of the sensorimotor coupling can take real values in the range of [-0.38,0.38] and the basic velocity ranges from 5cm/s to 10cm/s in steps of 0.2 cm/s. The transmission weights and v_0 are encoded with 8 bits each. The length of the resulting bitstring is thus $4 \times 4\text{bits} + 1 \times 3\text{bits} + 3 \times 8\text{bits} = 43\text{bits}$. The crossover and mutation probabilities are set according to the condition $C_{++}M_{+}$. Individuals which bump into a wall do not receive zero fitness as in GA 1. This is in order to support individuals that collide with a wall at a later time step. Collision here is punished by dividing the fitness they received at the point of collision by a factor of 2. A populationsize of 50 and $n_c = 2.0$ are used.

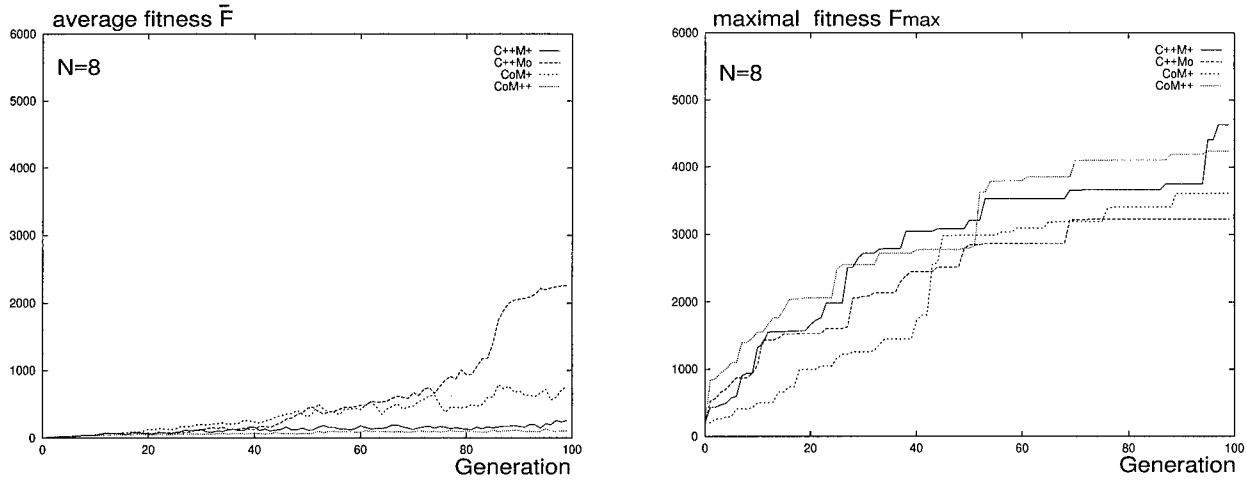


Figure 3: *Average (left) and maximal (right) fitness, averaged over 8 trials.*

4.4.1 Simulation block 1

The fitness function for the agent of type 2 in the simulation of block 1 is:

$$f = ksx_{max} \tag{8}$$

with $k = 1/2$ if the agent bumps into a wall and $k = 1$ if not. s is the length of the path the agent covers, and x_{max} the maximum value on the center axis of the tunnel the agent reached.

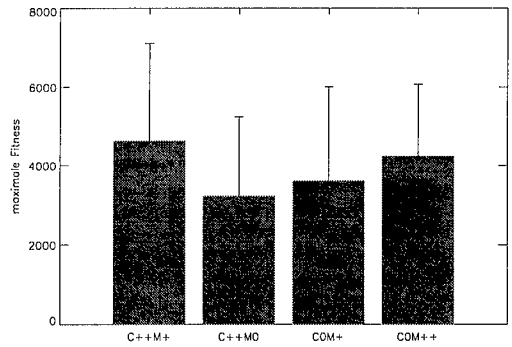


Figure 4: *GA 1: Maximal fitness at generation 100, averaged over 8 trials.*

4.4.2 Simulation block 2

Here the fitness function is:

$$f = k \sum_i |\Delta x_i| x_{max} \tag{9}$$

where $|\Delta x_i|$ is the distance on the center axis the agent covers in 10 steps. $|\Delta x_i|$ is computed every 10 steps and x_{max} is again the maximum value on the center axis of the tunnel the agent reached.

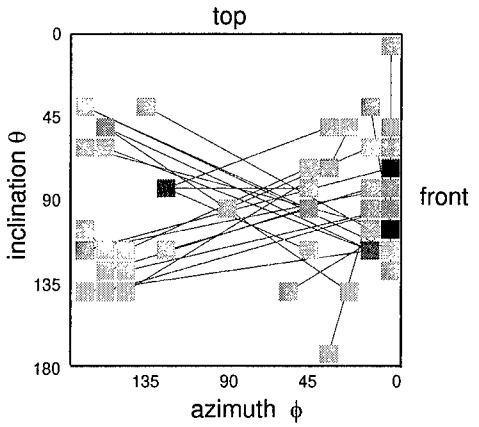


Figure 5: *Distribution of the orientations of the sensors for the best individual of each trial. The sensors forming a detector are connected by a line. The intensity of the sensors code the frequency of their occurrence, with darker greyvalues being more frequently.*

5 Simulations

5.1 Agent of type 1

In Fig. 3 the fitness $F_{\max}$ of the best individual and the average fitness $\bar{F}$ of the population for every generation, both averaged over 8 trials are shown. As a high proportion of the individuals bump into the wall and get zero fitness, the average fitness is much smaller than the maximum fitness for all four conditions. The maximal fitness after 100 generations averaged over 8 trials (see Fig. 3) is not significantly different between the different mutation and crossover probabilities. The average fitness $\bar{F}$ of the population is highest using only crossover followed by the $C_{++}M_{+}$ condition. The best individuals

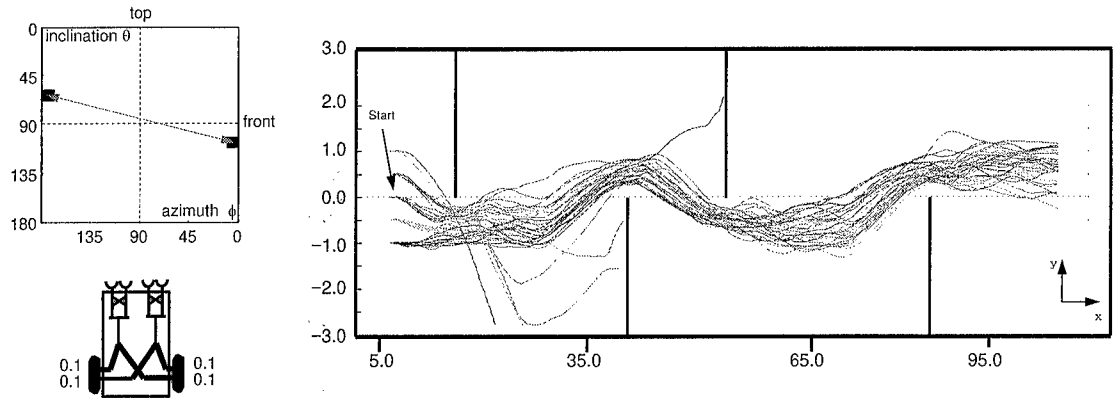


Figure 6: *Path in tunnel with 4 walls. Noise of* $\pm 10\%$ *is added to the input signals of the visual sensors and to the signals modulating the motor output. Even with different starting positions the agent travels through the tunnel in 93% of the trials successfully. Only in the extreme starting position* $y = 1.0$m *the agent bumps into the walls.*

are obtained with the $C_{++}M_{+}$ condition.

One might expect that searching randomly for the optimal solution is faster than applying a genetic algorithm to this problem. Evaluating the fitness of 10,000 randomly chosen individuals only 0.68% travel through the tunnel without colliding with walls. The maximal fitness that is found with this technique is F=1310. This indicates that every 150th individual has a fairly well fitness when using the random search technique. With genetic algorithms a much higher fitness up to F=9331 is found. The orientations of the sensors show a high variability (Fig. 5). Most of the agents evolve one sensor of each detector oriented in the heading direction, thus perceiving obstacles and the other sensor oriented towards the floor or ceiling of the tunnel or opposite to the heading direction. As long as there is no obstacle, correlating the two lowpass filtered sensor inputs leads to a symmetric detector output. With an obstacle detected in the front sensors an avoiding behavior is executed. The simulations are carried out with parameters for the mutation and crossover probabilities as described in Table 1. The motion detectors of the agent in Fig. 6 are oriented diagonal on the view sphere. One of the two sensors forming a detector on one hemisphere is oriented in the heading direction, the other backwards with an angular distance of 170°. The temporal change in the rotation angle here is $\dot{\varphi} = 0.2(d_L + d_R)/c$ deg/s. The velocity in the heading direction is $V = v_0$. Without obstacles the detector outputs are of equal magnitude, and due to the symmetric sensorimotor coupling the agent follows a straight line. A difference in the output of the movement detectors occurs if (i) the agent is not aligned to the center axis of the tunnel which leads to small turning reactions, (ii) obstacles appear in the field of view of the front sensor or (iii) the back sensor, with (ii) and (iii) causing large turning reactions in opposite directions. These three parts of the behavior enable the agent to avoid obstacles, starting with a large turning reaction if the obstacle appears in the front sensor (ii) and aligning itself back to the center line of the tunnel (i) which is supported by (iii), when the obstacle appears in the back sensor.

This architecture enables the system to generalize the behavior to unknown environments. Here instead of two, four walls with different distances are used. They are placed at $x = 15.0\text{m}, 55.0\text{m}$, $0.0\text{m} \leq y \leq 3.0\text{m}$ and $x = 40.0\text{m}, 85.0\text{m}$, $-3.0\text{m} \leq y \leq 0.0\text{m}$. In order to test how robust this agent is, noise of $\pm 10\%$ was added to the visual input and to the signals modulating the motor output. Under this condition even with different starting positions ($y = -1.0, -0.5, 0.0, 0.5$cm) the obstacles are avoided in 93% of the trials successfully (Fig. 6). Only in the extreme starting position $y = 1.0$cm the agent bumps into the walls.

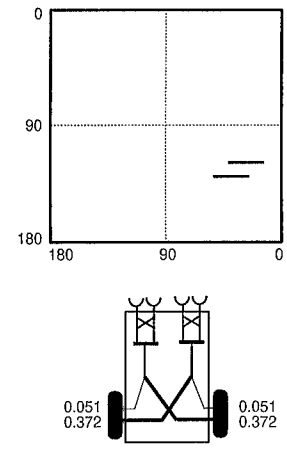


Figure 8: *The orientations of the two sensors forming a motion detector (the other two sensors are oriented symmetrically on the other hemisphere) and sensorimotor coupling for agent of type 2 resulting from the simulation of block 1.*

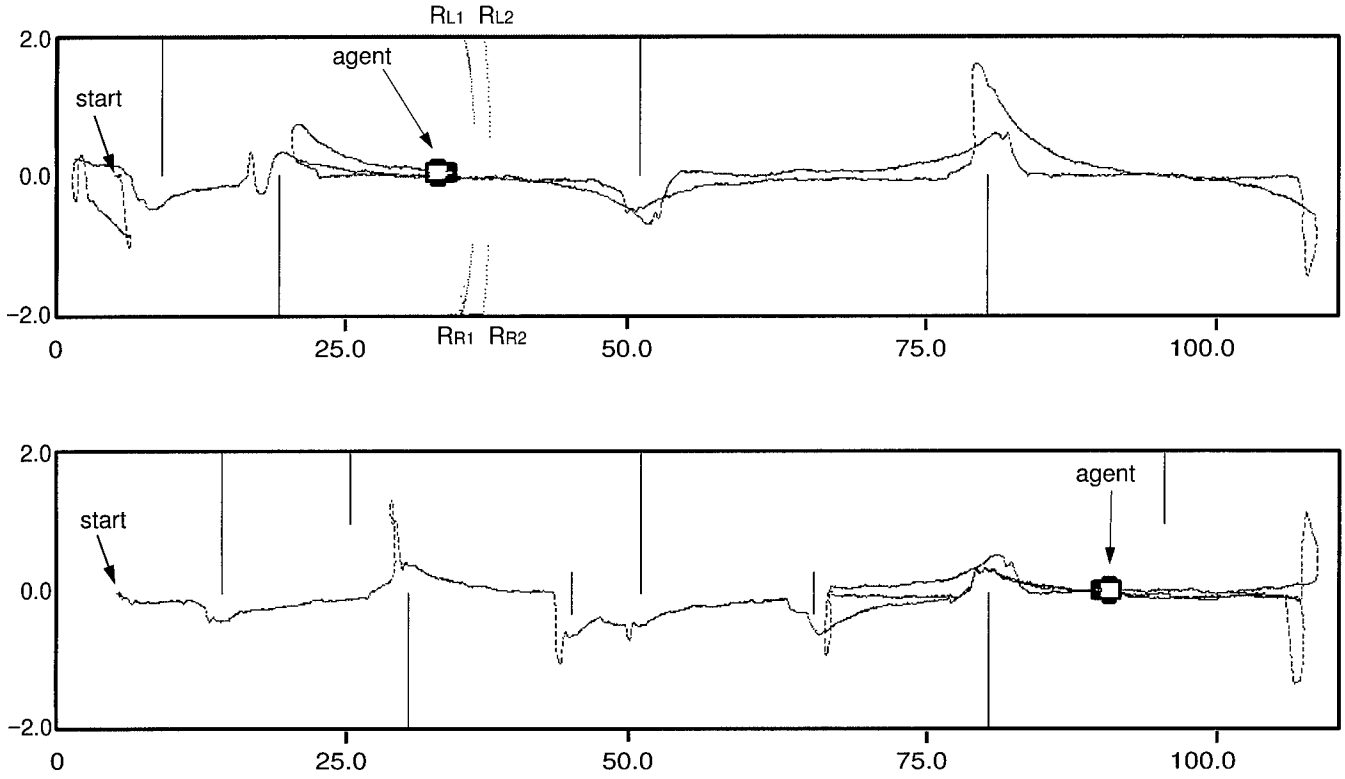


Figure 7: *Path in tunnel 1 (top) and tunnel 2 (bottom). R_{R1}, R_{R2} and R_{L1}, R_{L2} in tunnel 1 indicate the lines of intersection of the sensor rays with the walls of the tunnel. Noise of $\pm 10\%$ is added to the input signals of the visual sensors and to the motor output. Sinusoidal pattern is mapped onto the walls ceiling and floor ($\lambda = 2$).*

5.2 Agent of type 2

5.2.1 Simulation block 1

The agent of type 2 evolved in the simulation of block 1 (see Fig. 8) with the optical axes of its sensor orientations at 27.5° and 38.8° azimuth and 118.3° and 129.7° inclination. The evolved angular aperture of the sensors is with 27.5° much larger than for agent 1. The basic velocity of the motors is $\mathbf{v}_0 = 6.2$cm/s. The resulting angluar velocity is $\dot{\varphi} = -0.43(d_L + d_R)/c$ deg/s and the velocity in the heading direction $V = v_0 + 0.16(d_L - d_R)$ cm/s. With this architecture the sensors of an agent moving on the center axis receive visual input mainly from the floor and from a small part of the side walls. If the agent approaches an obstacle, response is smaller for the motion detector nearer to the obstacle. As the preferred direction of the motion detectors is almost vertical and the sinusoidal pattern is oriented vertically the walls and horizontally on the floor, the change of intensity in the sensors and thus the perceived motion decreases as the agent approaches an obstacle. The transmission weights for the contralateral connections are stronger than for the ipsilateral connections, hence the reduction of the detector output has a stronger effect on the velocity of the motor on the contralateral side and the agent turns away from the obstacle. The agents are tested in two different tunnels. Tunnel 1 is the original environment the agent evolved in, tunnel 2 differs in the number and position of the obstacles. Here we use 8 walls which are placed according to Fig. 7 (bottom). The test–trials are run with additional noise on sensor input and motor output. Table 2 describes the different noise conditions.
The agent has to survive 5000 time steps in the tunnel without bumping into a wall in order to show a successful behavior. With no additional noise the agent travels both tunnels in 100% of the time successfully. Adding more and more noise leads to a gradual reduction of performance. For tunnel 1 with $\pm 1\%$ noise, up to 70%–68% of the trials are successful, with $\pm 5\%$ 58–53 % and with a high noise of 10% still 46%–38% do not bump into a wall during 5000 time steps. For the tunnel 2 the performance is reduced to 44%–31% for the different conditions. In Fig. 7 examples of a successful travel through the tunnel 1 and tunnel 2 with 10% noise on sensor input and motor output are shown.

5.2.2 Simulation block 2

Here the agent evolves two sensors with the same inclination and overlapping receptive fields (Fig. 10). The optical axes are at 28° and 17° azimuth and 106° inclina-

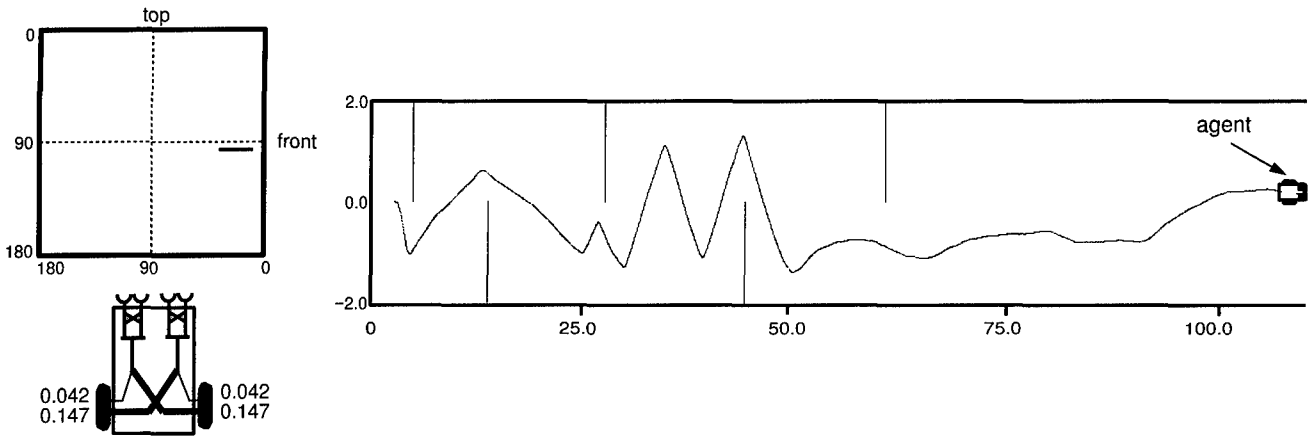


Figure 10: *The two sensors for agent 2 resulting from the simulation of block 2 on one hemisphere have overlapping receptive fields. Here their joint receptive field (top left) the transmission weights (bottom left) and the path in tunnel with 4 walls (right) are shown. A random dot pattern is mapped onto the walls, floor and ceiling.*

Condition	noise added to sensor input	noise added to motor output
s1	±1%	–
s5	±5%	–
s10	±10%	–
m1	–	±1%
m5	–	±5%
m10	–	±10%
s1m1	±1%	±1%
s5m5	±5%	±5%
s10m10	±10%	±10%

Table 2: *Conditions for noise testing.*

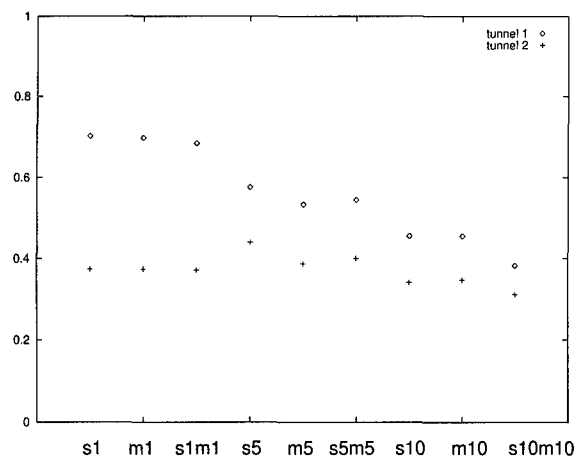


Figure 9: *Percentage of trials the individual of Fig. 8 travels tunnel 1 and tunnel 2 successfully. Noise according to table 2 is added.*

tion. The angular aperture of one sensor is 15°. Covering the region between 9.5° and 35.5°. The basic velocity evolved to $v_0 = 6.4$cm/s resulting in an angular velocity $\dot{\varphi} = -0.19(d_L + d_R)/c$ deg/s and the velocity in the heading direction $V = v_0 + 0.05(d_L - d_R)$ cm/s. Here the preferred direction of the motion detectors is horizontal and thus with a vertical sinusoidal pattern on the walls the response of the motion detector is larger on the side where the obstacle is detected. Again the transmission weights for the contralateral connections are stronger than for the ipsilateral. Here the detected motion has opposite sign compared to the agent of block 1 and thus the velocity of the motor on the side contralateral to the obstacle is reduced and a turning movement away from the obstacle results. Test–trials with additional noise do not show a stable behavior. This might be due to the fact that a rotation of the agent caused by noise has a much higher influence on a motion detector system that has a preferred horizontal direction than on on a detector system oriented vertically, as rotations around the vertical axis cause horizontal image flow but leave vertical image flow unchanged. This has to be investigated in further experiments.

6 Summary and future work

Autonomous agents adapted to the tasks of obstacle avoidance behavior during a simulated evolution using genetic algorithms. The agents develop the viewing direction of their sensors and the sensorimotor–coupling in a closed loop and are thus able to compensate for deviations caused by external disturbances and to avoid obstacles in different environments. The influence of the crossover and mutation probabilities on the outcome of the simulations, concerning the maximum fitness and the convergence of the population was tested. In this experimental setup the average fitness of the population is

highest if only crossover is used. Comparing the average maximal fitnesses obtained after 100 generations the use of crossover and/or mutation leads to comparable optimization results.

In future work we will evolve agents navigating in more complex environments. We plan to increase the number of movement detectors and use an array of sensors forming a 360° field of view. The agent will evaluate the motion detected in this field of view with filters that respond maximal to certain motion patterns – e.g. rotation around the vertical axis and translation in the direction of heading. Those filters are derived from the tangential neurons (see sect. 2) found in the visual system of the fly's brain. In addition the agents will receive more degrees of freedom making 3D flight manoeuvers possible.

7 References

[1] V. Braitenberg. Vehicles – experiments in synthetic psychology. The MIT Press, Cambridge, MA, 1984.

[2] R. A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, RA-2: 14–23, 1986.

[3] L. Davis (ed.). Handbook of genetic algorithms. Van Nostrand Reinhold, New York, 1991.

[4] M. Egelhaaf, A. Borst. Motion computation and visual orientation in flies. *Comparative Biochemical Physiology*, 104 A/4: 659–673, 1993.

[5] A. M. Flynn, R. A. Brooks. Battling Reality. *MIT AI Memo 1148*, 1989.

[6] J. D. Foley, A. van Dam, S. K. Feiner, J. F. Hughes. Computer Graphics – principles and practice, Addison–Wesley, 1987.

[7] S. Forrest. Genetic Algorithms: Principles of natural selection applied to computation. *Science*, 261: 872–878, 1993.

[8] N. Franceschini, J. M. Pichon, C. Blanes. From insect vision to robot vision. *Philosophical Transactions of the Royal Society of London* B 337: 283–294, 1992.

[9] K. G. Götz. Optomotorische Untersuchung des visuellen Systems einiger Augenmutanten der Fruchtfliege *Drosophila*. *Kybernetik*, 2: 77–92, 1964.

[10] D. E. Goldberg. Genetic algorithms in search, optimization, and machine learning. Addison Wesley, Reading, Mass, 1989.

[11] I. Harvey, P. Husbands, D. Cliff. Seeing the light: Artificial evolution, real vision. *Proccedingsof the 3rd International Conference on Simulation of Adaptive Behavior*, 392–401, 1994.

[12] B. Hassenstein, W. Reichardt. Systemtheoretische Analyse der Zeit-, Reihenfolgen- und Vorzeichenauswertung bei der Bewegungsperzeption des Rüsselkäfers *Chlorophanus*. *Zeitschrift für Naturforschung*, 11b: 513–524, 1956.

[13] K. Hausen. Motion sensitive interneurons in the optomotor system of the fly. I. The horizontal cells: structure and signal. *Biological Cybernetics*. 45: 143–156, 1982.

[14] R. Hengstenberg, H. Krapp, B. Hengstenberg. Visual sensation of self-motions in the blowfly Calliphora. In: *Biocybernetics of Vision: Integrative and Cognitive Processes* (ed.) C. Taddei–Ferretti (in press).

[15] J. H. Holland. Adaptation in natural and artificial systems. The University of Michigan Press, Ann Arbor, 1975.

[16] H. C. Longuet–Higgins, K. Prazdny. The interpretation of a moving retinal image. *Proceedings of the Royal Society of London* B 208: 385–397, 1980.

[17] H. Mühlenbein. Evolution in time and space – the parallel genetic algorithm. *Foundations of Genetic Algorithms*, Morgan Kaufmann, San Mateo, CA, 1: 316 –337, 1991.

[18] W. Reichardt. Autocorrelation, a principle for the evaluation of sensory information by the central nervous system. In *Sensory Communication*: 303–317, ed: W. A. Rosenblith, The MIT Press and John Wiley and Sons, New York, 1961.

[19] W. Reichardt. Functional characterization of neural interactions through an analysis of behavior. In *The neurosciences, fourth study program*: 81–104, ed: F. O. Schmitt, The MIT Press, London, 1979.

[20] W. M. Spears. Crossover or Mutation? *Foundations of Genetic Algorithms*, Morgan Kaufmann, San Mateo, CA, 2:221–238, 1993.

[21] N. Tinbergen. Instinktlehre – vergleichende Erforschung angeborenen Verhaltens. Parey Verlag, Berlin, Hamburg, 1953.

[22] H. Wagner. Flight performance and visual control of flight of the free–flying housfly (*Musca domestica L.*), III. Interactions between angular movement induced by wide– and smallfield stimuli. *Philosophical Transactions of the Royal Society of London*, B 312: 581–595, 1986.

[23] C. Wehrhahn, T. Poggio, H. Bülthoff. Tracking and chasing in houseflies (*Musca*). *Biological Cybernetics*, 45: 123–130, 1982.

The Evolutionary Cost of Learning

Giles Mayley
School of Cognitive and Computing Sciences,
University of Sussex, Brighton, BN1 9QH, England.
email: gilesm@cogs.susx.ac.uk

Abstract

Traits that are acquired by members of an evolving population during their lifetime, through adaptive processes such as learning, can become genetically specified in later generations. Thus there is a change in the level of learning in the population over evolutionary time. This paper explores the idea that as well as the benefits to be gained from learning, there may also be costs to be paid for the ability to learn. It is these costs that supply the selection pressure for the genetic assimilation *of acquired traits. Two models are presented that attempt to illustrate this assertion. The first uses Kauffman's NK fitness landscapes to show the effect that both explicit and implicit costs have on the assimilation of learnt traits. A characteristic 'hump' is observed in the graph of the level of plasticity in the population showing that learning is first selected for and then against as evolution progresses. The second model is a practical example in which neural network controllers are evolved for a small mobile robot. Results from this experiment also show the hump.*

1 Introduction

Biology is full of examples of situations where individuals learn and adapt to their environment during their lifetimes. The ability to adapt has come about through evolution to better the survival chances of those exhibiting it. Researchers have begun to use the a process of *artificial* evolution to produce artificial agents, including robots, that show similar adaptive qualities. However, learning and evolution interact in many ways. In 1896, J.M. Baldwin put forward a 'new factor' in evolutionary theory that has subsequently become known as the Baldwin Effect (Baldwin, 1896). He introduced what he called *organic selection* in an attempt to explain apparent examples of characteristics that were initially acquired by members of an evolving population, during their lifetimes, then becoming genetically specified in subsequent generations, without recourse to Lamarckianism. Organic selection is a process whereby individuals that are able to acquire a beneficial trait during their lifetime will be selected for, and so the ability to acquire that trait will be passed on to subsequent generations. Once the population is full of individuals that are able to acquire the trait, then that trait may become genetically specified or *assimilated* (Waddington, 1942) in later generations.

Recently, researchers have applied Artificial Evolution techniques to the study of the interactions between learning and evolution (Hinton & Nowlan, 1987; Whitley, Scott Gordon, & Mathias, 1994; Ackley & Littman, 1991). The emphasis of this body of work has been on looking at the advantages, in terms of performance, that the interaction gives evolution. Other researchers such as Todd and Miller (1991) and Stephens (1991) have applied artificial life techniques to investigate the level of predictability required in a changeable environment to support the evolution of learning. Here, I look at a different angle. During an evolutionary sequence in which genetic assimilation occurs there is a change in the level of learning that is present in the population. Belew (1989) and subsequently Harvey (1993) have analysed this change for the specific case of the model presented by Hinton and Nowlan (1987). A more general mechanism is described here that uses the idea that as well as the benefits that learning can bestow on an individual, there may also be costs to be paid for that learning ability. It is these costs and benefits that supply the selection pressure for the changes in the level of learning that is present in an evolving population.

The process of genetic assimilation will first be described and an evolutionary framework suggested in which it can take place. Next, the benefits and costs of learning will be described, along with the evolutionary trade-off between them that results in the genetic assimilation of acquired characteristics. Two very different evolutionary models will be presented that adhere to the evolutionary framework and that show genetic assimilation. The first uses NK fitness landscapes (Kauffman, 1993) to produce an abstract model for comparison with the second which involves the evolution of a controller for a real world robot.

It is assumed that the reader is familiar with the concepts of the genetic algorithm (G.A.) (Goldberg, 1989), evolutionary robotics (E.R.) (Harvey, Husbands, & Cliff, 1993) and artificial neural networks (N.N.) (Rumelhart & McCelland, 1986).

2 Genetic Assimilation

We start with a description of the classic example of genetic assimilation from biology, the ostrich callosities (Waddington, 1942; Maynard Smith, 1993). The skin of most vertebrates becomes tough and horny when subject to abrasion; i.e. it grows calluses. This growth is an adaptive trait since it has been evolutionarily selected for as a beneficial response to an environmental stimulus. Ostriches sit on the hot, hard ground and therefore have calluses on their rumps. However, it has been observed that these callosities appear on the foetuses whilst they are still within the egg and so cannot be the result of abrasion but have been genetically specified. The question is, how does the ostrich's developmental processes 'know' to put callosities on its rump before any environmental stimulus. It is fairly safe to assume that there was a time in evolutionary history when adult 'proto-ostrich' started sitting down for some reason, be it to make itself less conspicuous to predators, to avoid strong winds, or whatever. This would cause calluses to grow to prevent sores developing on its posterior. Individuals that could grow calluses earlier and therefore spend more time safely sitting down would have a selective advantage. The growth of calluses would appear earlier and earlier in the ostrich's lifetime over generations, requiring less and less environmental stimulus to produce them until no stimulus was required and the calluses appear on the foetus. They have thus become genetically specified through the process of genetic assimilation. It is clear that the selective pressure for growing callosities as early as possible comes from the disadvantage that having sores on the rump gives.

We will now look at this process in a more abstract way. Figure 1 shows an abstract behaviour space in which the individuals can learn behaviour A. Each point

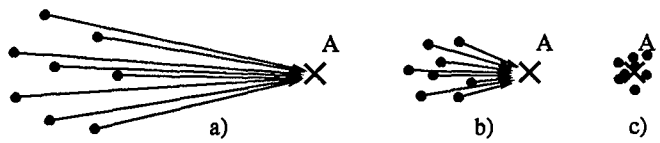


Figure 1: *Genetic Assimilation: the population moves from a) through b) to c) in a reduction of the amount of learning that takes place.*

represents the genetically specified behaviour of an individual and the amount of learning it undergoes is given by the length of the arrow from it to A. During the process of genetic assimilation, we would expect the population to move through a sequence something like that shown in Figure 1a, b, c. However, since all individuals can learn the good behaviour A in a), what is the selection pressure for this movement? That is, why is learning being selected against?

3 Evolutionary Framework

This section describes the evolutionary context of the ideas presented here. In this paper, no distinction is made between developmental adaptations, acquired characteristics and lifetime learning. All these terms come under the blanket term lifetime adaptation and will be used interchangeably, with an arbitrary preference for the term 'learning'.

3.1 The Goals of Evolution and Learning

In this work, since it is the adoption of learnt traits and their genetic assimilation that is under study, it is assumed that evolution and learning are operating to maximise the same fitness measure. If individuals in an evolving population are learning traits that are of no evolutionary worth, then it is unlikely that these traits and their subsequent genetic specification will be selected for. The point is made by Johnston (1982) that this is the case in biological evolution. Indeed, evolution seems to have gone to a lot of trouble to provide organisms with the appropriate pleasure and pain sensory systems to provide the reinforcement signals for it to be the case (it is these pleasure systems that we humans now abuse by taking drugs and eating too much chocolate). In an A-life context Ackley and Littman (1991) showed, through their use of evolved reinforcement learning that it is easier to evolve an evaluation function that constrains a learning network to perform an evolutionarily beneficial task than it is to evolve a good learning network alone. That is, evolution specifies the learning task that is beneficial to it. However, it is possible in artificial evolution to learn and evolve separate tasks as in Nolfi, Elman, and Parisi (1994). This is not the case here. In the case of the abstract model, the learning specifically maximises the same fitness measure whilst it is shown that learning is selected to increase fitness in the robotics model.

3.2 Genetically Specified Learning

A criterion that is important to the arguments that are presented here is that the ability to learn is under genetic control. That is, there exists a genetic mechanism that can specify whether a particular phenotypic trait is plastic or not such that evolution has a method of 'turning off' learning. This may be as simple as a genetically determined learning rate of zero in a N.N. Also, so that evolution has something to 'replace' the learning with, there should be a way of genetically expressing the characteristics of that trait in a fixed manner. These conditions are necessary for the process of genetic assimilation.

3.3 Combining Evolution and Learning

This section describes two ways in which learning and evolution have been combined. The subtle difference between the schemes that I want to highlight here is the way in which the fitness is attributed to each individual. In systems in which the learning algorithm is supervised, the learning is split into two distinct phases: a training phase and a testing phase. The fitness of each individual is evaluated *after* it has learnt on a training set by applying it to a test set and scoring it on how well it performs. Fitness is awarded after the individual's 'learning lifetime', or what I will call *posthumously*. An example of this involves the evolution of back-propagation neural networks as in (Chalmers, 1990). In unsupervised systems the training and testing occur concurrently. The individual learns and adapts during its lifetime and is scored continually throughout this process. Its final fitness score is then the accumulation of how well it performed the desired task from the moment it was 'born' until the end of its lifetime trial. The individual's ability to learn is under evaluation in the fitness trial as well as its performance in the given task after the learning process is complete. That is, the individual is under *continual assessment*. This is analogous to the situation that occurs in nature. In general, an individual does not get a chance to learn a trait without its actions, good or bad, affecting its evolutionary fitness.

The second classification, that of continual assessment, is of most interest here. It is argued in Section 4.2 that if this scheme is used then there are inherent costs to be paid for the ability to learn.

4 The Costs and Benefits of Learning

Johnston (1982) gives a full account of the costs and benefits of learning encountered in nature. Here I give a description of those costs and benefits that can be abstracted out to apply to both artificial and natural evolution [1].

4.1 The Benefits of Learning

This section describes some of the benefits that learning can give a population of evolving individuals.

The first is Baldwin's idea of organic selection in which he noted that evolution was almost able to 'predict' the direction in which to go. Adaptive members of the population are able to 'find' new advantageous behaviours that less plastic individuals are unable to perform. This means that these adaptive individuals gain the upper hand and are selected for. Thus evolution is guided by the actions of the individuals on which it operates. This process was first shown in a computational model, to great effect, by Hinton and Nowlan (1987).

Secondly, learning provides individuals with the ability to adapt to an environment that is changing at a faster time scale than that on which evolution operates (Johnston, 1982; Belew, 1989; Todd & Miller, 1991). Ackley and Littman (1991) go further to say that the combination of learning and evolution increases the spatiotemporal bandwidth of adaptability of a system thus providing the ability to cope with varied local spatial environmental differences as well as temporal ones. However, there are constraints on the nature of the variability as described by Stephens (1991). He states that there must be some degree of predictability to the variation for learning to be of any evolutionary use: "In an absolutely unpredictable environment, where today's state bears no predictable relationship to tomorrow's, then there is literally nothing to learn." Todd and Miller (1991) note that with this predictability, "learning may allow the genotype to 'store information in the environment' and let environmental regularities do much of the hard work of wiring up adaptive behavior-generators," This is another possible benefit of learning since more complex behaviour can be exhibited for a given genotype size.

Lastly, a learning mechanism may be able to provide an individual with behaviours that are simply very hard to evolve. For example, in humans, it is very difficult to imagine the English language being genetically specified at birth, even if the language were static and universal. It is better for evolution to provide an innate tendency to acquire a structured language and let learning sort out the details (Pinker, 1994).

4.2 The Costs of Learning

The previous section outlined some of the benefits that learning can give an evolving individual. It is probably less obvious that there are also evolutionary costs to be paid by an individual for that ability to learn.

First, using the assumption of Section 3.1, learning means the individual increases its fitness in some way during its lifetime. This implies that the individual spends some period of its lifetime performing at a lower level than that at which it is capable. Since we are looking at the final fitness score that an individual achieves as an accumulation of the acquisition of small fitness increments (continual assessment), then during this period the individual is not adding as significantly as it might to its fitness, i.e. it is incurring a *time-wasting cost*. This is the cost that led to the genetic assimilation of the ostrich callosities in Section 2.

Second, a learning individual may be actually reducing its fitness by performing tasks inappropriately and accruing an *incorrect behaviour cost*. Since learned behaviour is determined by the environment, this cost can arise if a vital behaviour-defining stimulus is not encountered.

[1] Some of Johnston's costs and benefits are specific to a group of species such as 'increased parental investment in each offspring' — a cost that is not universal to all evolved learners.

Also if a 'trial and error' learning strategy is used, the 'error' can lead to damaging or pathological behaviour. This may have little significance, such as a bird eating a bad-tasting berry, but conversely may cause the individual irreparable damage, such as falling down a precipice (learning the hard way). Possible mechanisms for reducing these costs in nature are parental care and animal play behaviour (Fagen, 1981). Lots of juvenile animals practice the more dangerous behaviours, such as hunting or fighting, that they will require later as adults, whilst they are under the protective care of their parents. Thus motor skills and coordination are learnt 'off-line' in the sense that failure will not result in their death — the incorrect behaviour costs have been reduced.

Another evolutionary cost discussed by Johnston (1982) is the increased *genotypic complexity* that is required to express the regulatory processes involved in any structural changes that take place during learning. The evidence for this is indirect in biological systems but in artificial systems where learning is genetically specified this will be the case. This contrasts with what was said in Section 4.1 about learning providing the benefit of more complex behaviour for a given genotype length. The truth is that for a given system there will be some sort of direct trade-off between them.

Lastly there is the possibility that the process of learning can cost an individual energy that it could more usefully expend in other pursuits. An individual may waste energy as well as time performing a series of candidate behaviours before selecting the most appropriate one. This is a minor learning cost but may be of significance if, for example, robot battery power or nutritional supplies are at a premium.

As long as the individual is evaluated using the continual assessment scheme outlined in Section 3 then it will incur at least some of these costs. However, if the posthumous assessment learning scheme is used where the process of learning is not under evolutionary evaluation then these costs are excluded and the individuals only receive the benefits of learning.

One last point to note is that the costs are relative within an evolving population. If two individuals can both exhibit the same behaviour but one has it genetically specified whilst the other learns it, the one with the innate behaviour will score a higher fitness, and will therefore be selected for, since it does not go through the learning process and accrue the costs.

4.3 Cost / Benefit Trade-off

The inclusion of learning can both be beneficial and detrimental to the fitnesses of individuals in a population and therefore there is an evolutionary trade-off as to whether it is adopted. Since the assumption was made in Section 3 that the existence of learning is under genetic control, then evolution might select against it if the costs are too high.

We can use the idea of learning costs and benefits to follow the varying selection pressures for and against learning that are present in an evolutionary sequence. In any evolving population there is genetic variation. When learning is genetically specified then there will be a distribution of learning abilities in any given population. All other things being equal, individuals that can learn a behaviour that increases their fitness, relative to their conspecifics who do not exhibit that behaviour, will have the selective advantage. Of course, for this increase in fitness, the benefits of learning have to outweigh the costs. The learning individuals will come to dominate the population and so we can say the benefits have supplied the selection pressure for the adoption of a learnt behaviour by the population. Once the population is full of learning individuals, we have the situation shown in Figure 1a. Now, the same variation in the learning abilities that sparked-off the adoption of the learned trait means that some individuals are 'closer' to the new behaviour and therefore perform less learning to exhibit it. In this way, they therefore accrue less learning cost and have a selective advantage over those that perform more learning. The advantage thus switches to those individuals who are genetically closer to the improved behaviour and the population moves through the sequence in Figure 1 under the selection pressure of decreasing the cost of learning. If we had a suitable measure for the amount of learning in the population, we would expect it to show a 'hump' as learning is in turn selected for and then against as the behaviour becomes genetically assimilated.

In the sequence just above, to aid the description, each stage was disjoint. Of course, in reality the transitions between each stage in the process would be much more continuous.

We thus have a description of a possible mechanism that causes the change in the level of learning in a population as genetic assimilation takes place. We now go on to describe two experimental models that have been designed with the constraints of Section 3 in mind for the purpose of investigating this mechanism.

5 The Abstract Model

The point to note from the preceding sections is that if the fitnesses of learning individuals in an evolving population are calculated using the continual assessment scheme, then there will be costs to be paid for the ability to learn. Furthermore, these costs can provide a selection pressure against learning and the genetic assimilation of any acquired characteristics may take place.

This section describes a simple experimental model in which this happens. The model is similar to that presented by Hinton and Nowlan (1987). However, their single-spike fitness landscape was considered inappropriate for this study of the cost and benefit trade-off since it

was desirable to allow evolution an 'alternative solution' to learning. That is, a landscape was desired that provided many local optima for search such that evolution could adopt a genetically fixed solution at a lower maximum than one found by learning if the costs of learning were too high. Kauffman's NK fitness model (Kauffman, 1993) was chosen since it provides an arbitrary fitness landscape of specified ruggedness.

5.1 *The NK Fitness Model*

In this model, N refers to the length of a binary genotype and the value of K sets the level of epistasis by determining the dependence the partial fitness of a gene at location n has on the genes in a neighbourhood of K other locations. The neighbourhood may be the K locations nearest to n in the genotype or a set of K locations randomly picked from anywhere on the genotype. A series of N lookup tables are generated, one for each gene location in the genotype. Each table has 2^{K+1} random entries in the interval (0,1). The fitness, F_{NK}, of a particular genotype is calculated by $F_{NK} = \frac{1}{N}\sum_{n=1}^{N} f(n)$ where the partial fitness $f(n)$ is obtained from the nth lookup table using the values of the genes in location n and its neighbourhood as the lookup key. This is illustrated in Figure 2a. Thus when $K = 0$, each gene contributes independently to the overall fitness of the genotype and the landscape is smooth; when $K = (N-1)$ the fitness landscape is maximally rugged — the fitness of a particular genotype is random and bears no relation to its neighbours in genotype space.

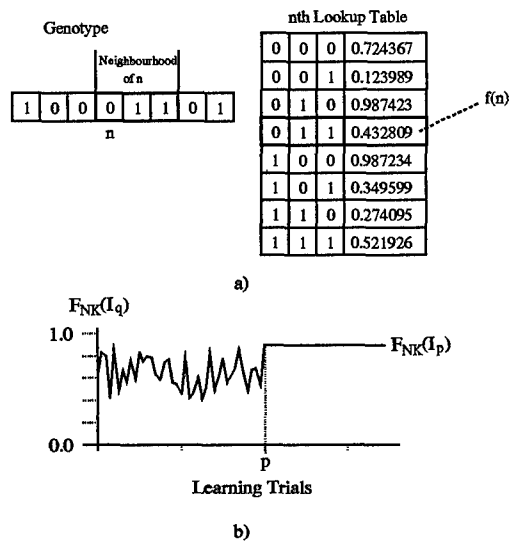


Figure 2: *a) Illustration of the calculation of $f(n)$ with $N = 8$, $K = 2$. b) Illustration of the scores obtained in a sequence of learning trials under continual assessment.*

As in the Hinton and Nowlan (1987) model, the genotypes in this experiment consist of strings of 20 genes containing one of three alleles: **0**, **1** or **?**. The phenotypes are also strings with 20 locations but containing **0**s and **1**s only. A **0** or **1** in the genotype translates directly into a **0** or **1** in the same position on the phenotype. These positions are considered genetically fixed. A **?** in the genotype indicates the corresponding position on the phenotype is plastic and may be set to **0** or **1** during the fitness evaluation by the learning procedure which is a simple random search. Each plastic location on the phenotype is randomly set to **0** or **1** with equal probability and the whole *phenotype* is then tested for its NK fitness (Note: the phenotype is now purely a binary string and the NK fitness model is applied to it, rather than to the genotype). This setting and testing is repeated 1000 times producing a sequence of 1000 different lifetime scores. We will say that the highest of these scores is achieved on the pth learning trial. An individual's fitness is calculated in one of two ways. In the first, an explicit cost of learning is imposed to provide a benchmark to see how the model behaves and also to show the effect that varying the cost can have on the amount of learning in the population and the performance of the G.A. in general. This is done using the fitness function:

$$F(i) = \frac{F_{NK}(I_p)}{F_{NK}(I_o)} - c\frac{m}{N} \qquad (1)$$

where $F_{NK}(I_p)$ is the highest fitness i achieved during its learning trials, $F_{NK}(I_o)$ is the global optimum for the landscape[2], $c\frac{m}{N}$ is the cost imposed on i for having m ?s in its genotype.

The second fitness function used here is designed to show that there is an implicit cost for learning if the members of the population are scored using the continually assessed learning scheme as mentioned in Section 3: no explicit cost of learning has been imposed:

$$F(i) = \frac{1}{F_{NK}(I_o)}\left(\left[\sum_{q=1}^{p-1}\frac{F_{NK}(I_q)}{1000}\right] + \left[\sum_{q=p}^{1000}\frac{F_{NK}(I_p)}{1000}\right]\right) \qquad (2)$$

where $F_{NK}(I_q)$ is the NK fitness achieved by i's phenotype, I, on the qth learning trial. Conceptually, the individual accumulates fitness score throughout its lifetime but it is able to 'recognise' when it achieves the best 'behaviour' that it is able to exhibit and undergoes no further learning for the rest of its lifetime (Figure 2b). Thus, the fitness score is the average of $p-1$ scores from the learning process of searching around the fitness landscape and $1000-(p-1)$ scores from the learnt behaviour.

The genotypes were evolved with a fairly standard genetic algorithm using a population of 1000, linear rank-based selection and unity crossover probability. The

[2]The global optimum is found using exhaustive search at the beginning of a program run. The fitness values are scaled by it to allow comparison between runs on different landscapes to be made. Since linear rank-based selection is used, this scaling has no effect on the G.A.

three alleles are distributed in the initial population with the probabilities $p(0)$, $p(1)$ and $p(?)$ respectively; all equal to $\frac{1}{3}$. The mutation rate was set at 0.05 mutations on average *per genotype* with equal probability of mutating to either of the other two alleles. A low mutation rate was used due to the short genotype length.

5.2 *Results of the Abstract Model*

This section presents the results from experiments carried out using the abstract model. All graphs are the mean of 30 evolutionary runs on landscapes with $N = 20$ and $K = 0, 5, 10, 15$ and 19, using nearest neighbours, with a different landscape being generated for each K in each run. First shown is evolution alone with no learning, then evolution with an explicit learning cost and finally learning with the implicit learning costs.

5.2.1 *Evolution Without Learning*

The graph in Figure 3a shows results from the G.A. operating on the NK fitness landscapes without learning, i.e. there are no ?s in the initial genotypes and a **0** or **1** cannot mutate to a **?**. As expected, evolution does less well on highly epistatic landscapes than smooth ones. This graph is included for comparison with the results in Sections 5.2.2 and 5.2.3.

5.2.2 *Explicit Learning Costs*

Two sets of results are shown in Figure 3 from runs that assign an explicit cost of learning by using Equation 1 to evaluate fitness. In the first set, Figures 3b and 3c, c is given a value of 0.05 — a low cost of learning. With $K = 0$, evolution alone performs better than evolution with learning, i.e. it reaches the optimum in less generations. This is to be expected since close phenotypes achieve similar scores on a smooth landscape and so the learning process, which is doing a local search, is not increasing fitness significantly (low benefits) whilst there are costs to be paid. It can be seen that learning disappears from the population quickly (Figure 3c). The difference comes when K is large. In the initial stages a learning population receives a large benefit since on a rugged landscape the population is always near areas of increased fitness but, because c is small, pays relatively little cost. Thus learning is initially selected for. As the population settles down on a local optimum, individuals that can achieve that optimum whilst accruing less learning cost are selected for and the proportion of ?s falls. Another way of looking at this is, with high K, for any particular locus, it is not defined whether or not a **1** or **0** would be more beneficial until the contents of the other loci in its neighbourhood have been set. It is therefore more advantageous to specify a **?** in the early stages of evolution. Later, as the contents of the neighbourhoods settle down, then the partial fitnesses of **1**s and **0**s become more defined and evolution is able to decide between them.

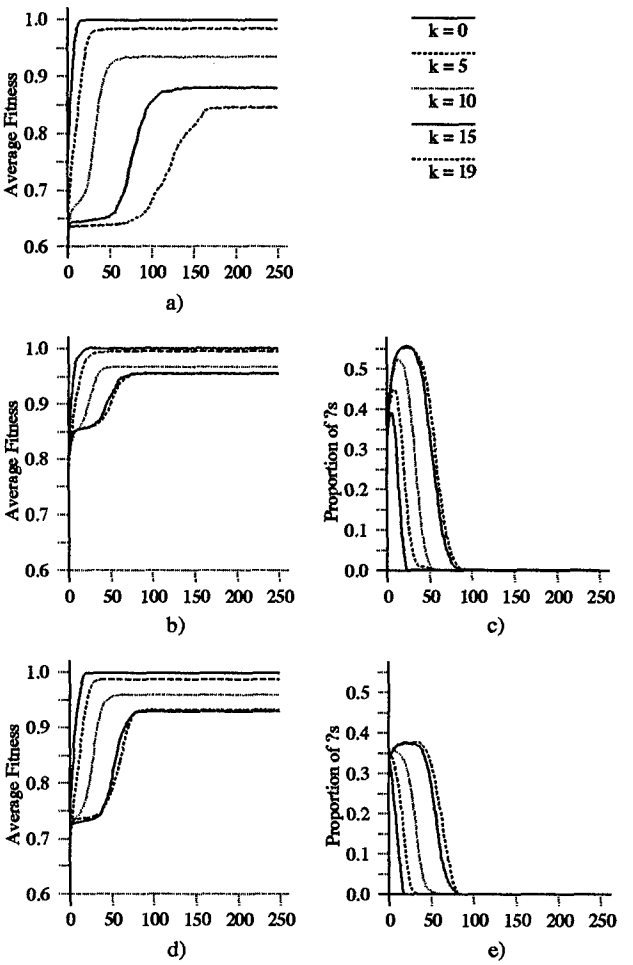


Figure 3: *a) Average fitness of members of the population without learning. b) Average fitness with explicit learning cost, $c = 0.05$. c) Average proportion of ?s with explicit learning cost, $c = 0.05$. d) Average fitness with explicit learning cost, $c = 0.3$. e) Average proportion of ?s with explicit learning cost, $c = 0.3$. All graphs are shown over generations.*

In Figures 3d and 3e, c is given a value of 0.3 — a high cost of learning. It can be seen that, for low K, the costs of learning heavily outweigh the benefits and the ?s are removed from the population in a few generations. As K increases, the learning does become advantageous but it is not as strongly selected for as when $c = 0.05$ because of the increased costs. The population reaches areas of higher fitness in the landscape than evolution alone but not as high as when the costs were lower.

5.2.3 *Implicit Learning Costs*

The graphs in Figure 4 show the results from a run

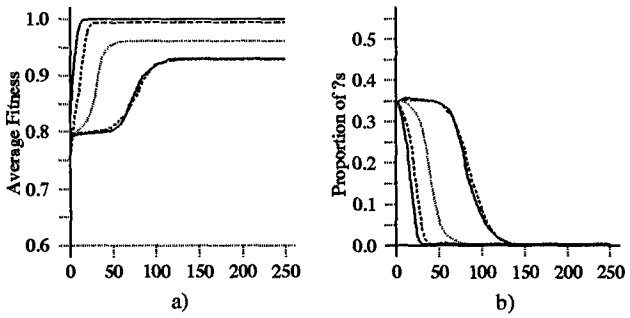


Figure 4: *a) Average fitness with implicit learning costs. b) Average proportion of ?s with implicit learning costs. Both graphs are shown over generations.*

where fitness is calculated using Equation 2. This corresponds to a situation where there is continual assessment of the fitness through the learning trial and any learning costs are implicit. Everything is as it was in Section 5.2.2 except that now the learning costs are implicit. We can see that there are similarities to the final average fitness values reached in the explicit experiment when $c = 0.3$ and in the proportion of ?s that appear in the gene pool over generations. This indicates in this case, continual assessment has led to a fairly high cost of learning.

A true comparison between Figures 3d and 3e, and Figures 4a and 4b is difficult because of the differences in the way that the overall cost varies with the number of ?s in the two evaluations. When the cost is set explicitly, it increases linearly with the number of ?s as determined by the $c\frac{m}{N}$ term in Equation 1. However, when the cost is implicit, it is proportional to the expected value of p, the number of trials taken to find the highest learning score, which varies exponentially with the number of ?s. This may suggest why the graphs for large K in Figure 4a, show a lower level of ?s being sustained in the population for more generations than in Figure 3e. That is, the cost associated with the present level of ?s may be fairly small with respect to the benefits but the increase in learning cost associated with an increase in ?s may prohibit their selection.

In general then, this model has shown us what happens to the level of plasticity in an evolving population over generations if learning is penalised by a cost. Furthermore, if the continual assessment scheme is used in which the learning is under genetic control then the cost is implicitly provided by the learning process itself.

6 Evolutionary Robotics Model

We turn now to practical example of genetic assimilation of acquired characteristics for the purpose of comparison with the results obtained in the abstract model presented in Section 5.

6.1 *Khepera*

The Khepera[3] (K-Team, 1993) is a small robot, 5.5cm in diameter, with 8 combined infra-red (I.R.) proximity detectors and ambient light sensors, and differential drive from two motorized wheels (Figure 5a). A neural network control structure is evolved for it using the Khepsim[4] simulator (Jakobi, Husbands, & Harvey, 1995) for the task of wall-following using only the I.R. sensors, which can detect objects to a distance of about 10cm. Since the behaviour of the robot in the simulator closely matches the behaviour in the real world, no distinction will be made between them. The environment is shown in Figure 5b along with the three starting positions for the three lifetime trials that each individual receives (see Section 6.3). The walls in the environment form angles that are less than 90°, greater than 90° and greater than 180°, all of which a successful controller must cope with.

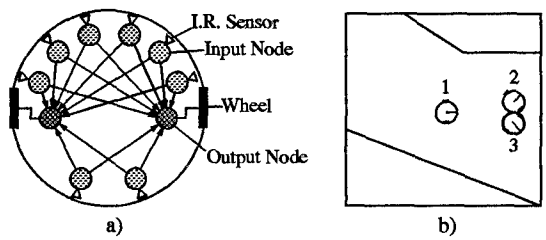


Figure 5: *a)The Khepera robot showing the neural net configuration. b) The environment showing the initial positions for an individual's three lifetime trials.*

6.2 *Nets 'n' Things*

The Khepera is controlled by a simple feed-forward neural network as shown in Figure 5a. There are 8 input nodes, one for each I.R. sensor, and 2 output nodes, one for each wheel. The net is fully connected and so there are 16 uni-directional connections from the input layer to the output layer, each of which has a weight associated with it. These weights can be either excitatory or inhibitory with values in the range (0,1) and are modified by the learning algorithm. The input nodes have linear activation functions which scale the I.R. sensor reading to between 0 and 1. If a_j is the activation of the jth input node then the activation of the ith output node, a_i, is calculated thus:

$$a_i = \frac{1}{3}\left(\sum_{j=0}^{7} a_j w_{ij}^E - a_j w_{ij}^I\right) + 0.5 \quad \begin{array}{l}\text{if } a_i < 0 \text{ then } a_i = 0 \\ \text{if } a_i > 1 \text{ then } a_i = 1\end{array} \tag{3}$$

[3] The Khepera robot was developed at EPFL, Lausanne by Francesco Mondada and colleagues.

[4] The Khepsim simulator was developed at School of Cognitive and Computing Sciences, University of Sussex by Nicholas Jakobi.

where w_{ij}^E or w_{ij}^I is the weight of the excitatory or inhibitory connection respectively from j to i and $\sum_{j=0}^{7} a_j w_{ij}^E - a_j w_{ij}^I$ is the net input to i. The activation function is thus a linear threshold. The output, o_i, from an output node to its motor is given by $o_i = 1.5a_i - 0.5$ such that when there is no net input to i, then the motor will turn forward at a quarter of its maximum speed; when $a_i = 1.0$ then the motor will go at full speed in the forward direction and when $a_i < \frac{1}{3}$, the motor is in reverse.

The network weights are directly encoded onto a binary genotype, each weight represented by a 5 bit segment whose position on the genotype signifies which weight it codes for. In each segment the first bit (`I/E`) represents whether the weight is inhibitory or excitatory, the next two bits (`init`) give the initial value, Gray coded to give the values 0.0, $\frac{1}{3}$, $\frac{2}{3}$ and 1.0. The fourth bit (`plas`) codes for whether the weight is plastic or not. If `plas` is set, then the weight is modified by the learning rule, if unset then the weight remains at the genetically specified initial value for the duration of the individual's lifetime trial. The fifth bit (`inc/dec`) defines the effect the learning rule has on the weight of the connection, i.e. whether or not the magnitude of the weight is increased or decreased during learning. With 16 connections the genotype is therefore 80 bits long.

During each individual's lifetime trial, the weights of the plastic connections are changed according to the learning rule. This takes the form of a simple Hebbian, with either a positive or negative learning rate for each weight depending on the state of the appropriate `inc/dec` bit on the genotype:

$$\Delta w_{ij} = \varepsilon_{ij}(a_j a_i) \tag{4}$$

where Δw_{ij} is the change in w_{ij}, ε_{ij} is the learning rate for w_{ij}, either 0.01 or -0.01. The weights are bounded to between 0.0 and 1.0 by simple thresholds. Note, it is only during the calculation of the net input to the output nodes in Equation 3 that the excitatory and inhibitory connections are treated any differently. The learning algorithm affects only the magnitude of the weights, e.g. an inhibitory weight will get less inhibitory with a negative learning rate.

6.3 Genetic Algorithm

A population of 30 individuals were evolved to exhibit wall-following behaviour in the environment shown in Figure 5b. A fairly standard G.A. was employed that used a crossover probability of 0.7, a mutation rate of 0.7 mutations on average *per genotype* and linear rank-based selection. Each individual receives 3 lifetime trials, one for each of the starting positions shown in Figure 5b, with its network reset to the genetically specified initial weights at the beginning of each trial. A trial lasts 1000 simulation steps, simulating 100 seconds of real time. The desired robot behaviour was that of travelling in a straight line, close to walls but without hitting them. The fitness function used to achieve that can colloquially described as "the distance travelled whilst the robot can 'see' a wall with one of its I.R. sensors minus the total change in the robot's orientation when it can 'see' a wall minus a penalty every time it hits a wall, all averaged over the number of iterations of the trial." The robot is judged to be able to see a wall if one or more of its sensors register over $\frac{1}{10}$th of their maximum value. The fitness score for one trial is thus:

$$F_T(I) = \frac{\sum_{t=0}^{1000} B_t \left(\sqrt{(x_t - x_{t-1})^2 + (y_t - y_{t-1})^2}\right)}{1000} - \frac{\sum_{t=0}^{1000} B_t \left(|\theta_t - \theta_{t-1}|\right)}{1000} - \frac{\kappa W}{1000} \tag{5}$$

where $B_t = 1$ if the robot can see a wall at time-step t, 0 otherwise; x_t, y_t are the x and y coordinates of the centre of the robot at time-step t; θ_t is the orientation of the robot at time t; κ is the crash penalty, set at 1.0 and W is the number of time-steps the robot was in contact with the walls. The overall fitness of the individual is calculated as the mean of the three trial fitnesses.

The G.A. was run for 300 generations, the results of which are shown in Section 6.4.

6.4 Preliminary Results

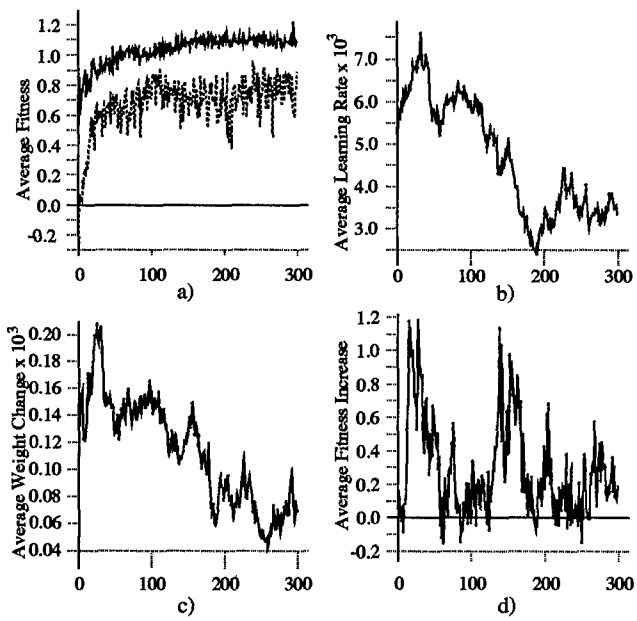


Figure 6: *a) The best and average fitness scores. b) Average learning rate per weight per individual. c) Average change in weight each iteration per connection per individual. d) Average fitness increase afforded by the learning process for each individual. All graphs are shown over generations.*

Figure 6 shows the results from a typical run of the Khepera robot model. Figure 6a shows the average fitness of the members of the population and the fitness of the best member of each population over generations. Figures 6b, 6c and 6d show three different measures for the amount of learning that occurs in the population as evolution progresses. The first is the average magnitude of the learning rate per weight per individual, where non-plastic connections are awarded a learning rate of zero. This measure represents the genetically specified potential for phenotypic plasticity that is present in the population over generations. The expected value of this measure for a random population is 0.005 and the graph spends a significant number of generations above this in the early stages of evolution before falling to well below.

Figure 6c shows the average change per iteration per connection per individual in the weight values during the learning trials. It represents the phenotypic change that actually occurred in each generation through evolutionary time. As we can see this graph also shows the characteristic 'hump' indicating that there is an increased level of learning in early generations.

The third measure (Figure 6d) shows the average increase in fitness that the learning process gives the population over generations. Each individual performs a fourth lifetime trial in which all its weights are clamped at their initial values for the duration of the trial. The score obtained from this trial, which is started at position 1 in Figure 5b, is subtracted from the fitness of the position 1 learning trial to give the increase in fitness. This fourth trial is performed solely to produce this measure and takes no part in the evolutionary process. Notably for almost the entire run, learning is providing a net fitness benefit to the population. It is almost certainly the case that there are possible genotypes in which learning is not beneficial but evolution has selected against these in the early stages. The Khepera model complies to the conditions laid down in Section 3 in that the learning is under genetic control and the fitness is calculated under continual assessment. The learning algorithm is beneficial to the evolutionary process and therefore we would expect there to be an increase in the level of learning at the beginning of an evolutionary run. This level should then fall in subsequent generations and, using the measures described above, this appears to be so. We thus have correspondence between the results obtained from the NK model and those from the Khepera model.

7 Discussion

This paper investigates the particular case in the rich area of the interaction between learning and evolution, where a learned trait or behaviour becomes genetically assimilated. A mechanism is suggested which uses the idea that as well as the benefits to be gained from learning, there may also costs to be paid by the members of an evolving population for that ability to learn. It is the exploitation of the benefits and reduction of the costs that provide the selection pressure for firstly the adoption of a learned behaviour and subsequently its genetic assimilation. Furthermore, if the fitness of the individuals are assessed using the continual assessment scheme, then there will always be costs.

Two models were presented that conformed to the constraints that were given in Section 3. The first was abstract and based around the NK fitness model (Kauffman, 1993). Here, two separate characterisations of the cost of learning were experimented with, one with the costs explicit, the other implicit through the use of a continually assessed fitness score. Both modes of operation led to the exploitation of learning followed by its genetic assimilation giving a characteristic 'hump' in the graphs showing the plasticity of the members of the population. Comparison between the results from the different modes showed that the cost of learning in the continually assessed case was reasonably high. Work is continuing to characterise the exact cost of learning associated with this scheme including the effect that the level of epistasis has on it.

The second model was a practical example using an evolutionary robotics methodology. The experimental setup conformed to the framework that was set up in Section 3 and the results compare well with those of the NK model — the characteristic hump appeared in the measurements of the level of learning in the population. It was confirmed that learning really was increasing the fitnesses of the members of the population by comparing one of their learning trials with a non-learning one.

It is felt that the simplicity of this model provides a bed for the tractable analysis of ideas such as those presented here, whilst still allowing the evolution of learning behaviours in the real world. The learning rule and genetic encoding of the weights and plasticity information provide a direct route for structural changes that were due to learning to become genetically specified. Future work will investigate ways in which the specific learning costs that were described in Section 4.2 can be varied in this model. For example, the incorrect behaviour costs can be changed by varying the crash penalty, κ, or the time wasting costs by changing the trial lengths.

In the longer term, it is hoped that the model can be used to investigate some of the issues raised by Todd and Miller (1991) and Stephens (1991) concerning the evolution of learning in predictably changing environments. A mechanism is already in place to provide environmental variability by changing the calibration and connectivity of the I.R. sensors on the robot.

It is hoped that the work here has shed a little more light on the subtle and counter-intuitive nature of the interactions between evolution and learning.

In conclusion, if one uses the continual assessment

scheme with a genetically specified ability to learn, the costs associated with learning provide the selection pressure for the genetic assimilation and learning may disappear from the population.

Acknowledgements

Many thanks to Inman Harvey, Adrian Thompson, and Dave Cliff for their help and contributions and thanks to my reviewers for their useful comments.

References

Ackley, D., & Littman, M. (1991). Interaction between learning and evolution. In Langton, C., Taylor, C., Farmer, J., & Rasmussen, S. (Eds.), *Artificial Life II*, pp. 487–509.

Baldwin, J. (1896). A new factor in evolution. *The American Naturalist*, pp. 441–451.

Belew, R. (1989). Evolution, learning and culture: Computational metaphores for adaptive algorithms. Tech. rep. CS89-156, Comp.Sci, Engr. Dept, Uni. California at San Diego.

Chalmers, D. (1990). The evolution of learning: An experiment in genetic connectionism. In Touretzky, D. S., Elman, J. L., Sejnowski, T. J., & Hinton, G. E. (Eds.), *Proceedings of the 1990 Connectionist Models Summer School*. Morgan Kaufmann.

Fagen, R. (1981). *Animal Play Behavior*. Oxford University Press.

Goldberg, D. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley.

Harvey, I. (1993). The puzzle of the persistent question marks: A case for genetic drift. In Forrest, S. (Ed.), *Genetic Algorithms: Proceedings of the Fifth International Conference (GA93)*. Morgan Kaufmann, San Mateo, CA.

Harvey, I., Husbands, P., & Cliff, D. (1993). Issues in evolutionary robotics. In Meyer, J., Roitblat, H., & Wilson, S. (Eds.), *From Animals to Animats 2: Proceedings of the Second International Conference on Simulation of Adaptive Behavior*. MIT Press.

Hinton, G., & Nowlan, S. (1987). How learning can guide evolution. *Complex Systems, 1*, 495 – 502.

Jakobi, N., Husbands, P., & Harvey, I. (1995). Noise and the reality gap: The use of simulation in evolutionary robotics. In *Proceedings of the 3rd European Conference on Artificial Life*. Springer-Verlag.

Johnston, T. (1982). Selective costs and benefits in the evolution of learning.. In Rosenblatt, J., Hinde, R., Beer, C., & Busnel, M. (Eds.), *Advances in the Study of Behavior.*, Vol. 12. Academic Press.

K-Team (1993). *Khepera Users Manual*. EPFL, Lausanne.

Kauffman, S. (1993). *The Origins of Order: Self-Organization and Selection in Evolution*. Oxford University Press.

Maynard Smith, J. (1993). *The Theory of Evolution* (3rd edition). Cambridge University Press.

Nolfi, S., Elman, J., & Parisi, D. (1994). Learning and evolution in neural networks. *Adaptive Behavior, 3*, 5–28.

Pinker, S. (1994). *The language instinct : the new science of language and mind*. Allen Lane.

Rumelhart, D., & McCelland, J. (1986). *Parallel Distributed Processing*, Vol. 1. MIT Press.

Stephens, D. (1991). Change, regularity and value in the evolution of animal learning. *Behavioral Ecology, 2*, 77–89.

Todd, P., & Miller, G. (1991). Exploring adaptive agency II: Simulating the evolution of associative learning.. In Meyer, J., & Wilson, S. (Eds.), *From Animals to Animats: Proceedings of the first International Conference on Simulation of Adaptive Behavior*, pp. 306–315. MIT Press.

Waddington, C. (1942). Canalization of development and the inheritance of acquired characters. *Nature*, pp. 563–565.

Whitley, D., Scott Gordon, V., & Mathias, K. (1994). Lamarckian evolution, the Baldwin Effect and functional optimization. In Davidor, Y., Schwefel, H. P., & Manner, R. (Eds.), *Parallel Problem Solving from Nature-PPSN III*, pp. 6–15. Springer-Verlag.

Evolving Obstacle Avoidance Behavior in a Robot Arm

David E. Moriarty and Risto Miikkulainen
Department of Computer Sciences
The University of Texas at Austin
Austin, TX 78712
moriarty,risto@cs.utexas.edu

Abstract

Existing approaches for learning to control a robot arm rely on supervised methods where correct behavior is explicitly given. It is difficult to learn to avoid obstacles using such methods, however, because examples of obstacle avoidance behavior are hard to generate. This paper presents an alternative approach that evolves neural network controllers through genetic algorithms. No input/output examples are necessary, since neuro-evolution learns from a single performance measurement over the entire task of grasping an object. The approach is tested in a simulation of the OSCAR-6 robot arm which receives both visual and sensory input. Neural networks evolved to effectively avoid obstacles at various locations to reach random target locations.

1 Introduction

Many industrial tasks such as assembly, packaging, and processing rely heavily on the manipulation and transportation of small components. Robot arms can automate many of these processes and improve the cost efficiency of the operation. To be as effective as their human counterparts, robot arms normally use vision systems based on one or more cameras to identify and locate the target objects (Feddema and Lee 1990; Papanikolopoulos and Khosla 1993; van der Smagt 1995; Weiss et al. 1987; Wijesoma et al. 1993).

Vision-based robot arm control is a very complex task that requires mapping the information of the target and obstacle locations to the joint rotations that position the hand near the target. This task in commonly known as hand-eye coordination. Because it is difficult to specify such a mapping by hand, many researchers have applied machine learning techniques to learn the control strategy. The resulting policies are often much more robust than manually-designed, fixed policies.

One particularly successful approach to control learning is *neuro-control* (Baker and Farrell 1992; Werbos 1992) or using a neural network to learn to implement the control policy. Most approaches to robot arm neuro-control learn hand-eye coordination through supervised training methods such as backpropagation or conjugate gradient descent (Kawato 1990; Miller 1989; van der Smagt 1995; Werbos 1992). Supervised learning, however, requires training examples that demonstrate correct mappings from input to output. Training examples are normally generated by flailing the arm while recording joint movements and final arm positions (Werbos 1992). The supervised approach is sufficient for learning basic hand-eye coordination in domains with unrestricted movement, however, in uncertain or obstacle-filled domains the supervised approach fails. The main problem is that in the supervised training the arm is always moved to the target location in a single joint rotation. In domains with obstacles, this is not always possible because the arm must move around an obstacle. Explicitly generating the necessary intermediate joint positions is extremely difficult and requires significant domain knowledge (Lumelsky 1987). Without such knowledge, it is not possible to learn obstacle avoidance behaviors through supervised methods.

This paper presents an alternative learning control system that *evolves* neural networks through genetic algorithms. Neuro-evolution does not require explicit training examples and learns multiple joint rotations implicitly through evolution. The evolutionary search is guided by a single fitness evaluation over the entire task, which may involve multiple skills such as avoiding obstacles and reaching the target. Neuro-evolution can thus form network controllers that can adapt in uncertain environments.

Evolving neuro-control was tested in a sophisticated robot arm simulation of the OSCAR-6 anthromoporphic robot. The evolution was based on the Hierarchical SANE system (Moriarty and Miikkulainen 1996a, 1996b). Networks were evolved to maneuver the arm to random target locations while avoiding obstacles. Given both camera-based visual and infrared sensory input, the networks learned to effectively combine both target reaching and obstacle avoidance strategies. The current experiments detect obstacle collisions only in the robot's hand; future experiments will extend the control policy

to avoid obstacles with the full arm.

The body of this paper is organized as follows. The next section summarizes the existing methods for learning robot arm control in neural networks and describes their limitations. Section 3 motivates the use of neuro-evolution, and section 4 describes the implementation of the Hierarchical SANE system to evolve hand-eye coordination and obstacle avoidance in the OSCAR-6 robot. Section 5 describes the experiments conducted in an OSCAR-6 simulator and summarizes the main results. Enhancements to the simulator and applications to real robot arms are outlined in section 6, followed by our conclusions in the final section.

2 Learning to Manipulate a Robot Arm

Several methods have been proposed for learning robot arm control in neural networks (Kawato 1990; Kuperstein 1991; Miller 1989; van der Smagt 1995; Walter et al. 1991). Each of these methods is based on supervised learning from a corpus of input/output examples that demonstrate correct behavior. During training, the network is presented inputs from the database and its output is compared to the desired output. Errors are calculated according to the differences, and modifications are made to the network's weights based on some variant of the backpropagation algorithm. In any supervised learning application, it is crucial that the training corpus contains a good representative sample of the desired behavior.

The most common approach for generating training examples is to flail the arm and record the resulting joint and hand positions (Werbos 1992). For example, if the joints are initially in position $\vec{J}$ and a random rotation $\vec{R}$ results in hand position $\vec{H}$, a training example of the form $Input : \vec{J}, \vec{H}; Output : \vec{R}$ can be constructed. This example reflects the correct rotation to reach target position $\vec{H}$ from joint position $\vec{J}$. Given a sufficient database of such examples, a neural network can learn to approximate the inverse kinematics necessary to translate between the camera-based visual and joint spaces.

A major limitation of generating training examples by "flailing" is that it only applies to situations where the target can be reached in a *single* joint rotation. It cannot demonstrate more general behavior such as reaching while simultaneously avoiding obstacles, where a sequence of rotations are necessary. For example, when an obstacle is placed between the arm and the target, the arm cannot take a direct path, but must instead make several moves around the obstacle. Random arm movement would never produce a sufficient training example for this situation, since there is no single rotation that can reach the target. To produce such behavior using a supervised learning approach, training examples must demonstrate movement to intermediate arm positions (e.g. above the block). It is unclear how such examples could be generated without a path-planning algorithm (Lumelsky 1987), which requires significant domain knowledge of the robot and its environment.

An alternative to supervised learning is to use a *reinforcement learning* method, such as Q-learning (Watkins and Dayan 1992) or genetic algorithms (Goldberg 1989), to form the control policy. In reinforcement learning, no input/output examples are required, and thus no path planning algorithms are necessary to generate intermediate arm positions. Agents learn from signals that provide some measure of performance and which may be delivered after a sequence of decisions have been made. Since reinforcement signals take into account several control decisions at once, appropriate credit can be assigned to the intermediate joint rotations that are necessary to reach the final target position. Reinforcement learning can thus integrate sophisticated behaviors such as obstacle avoidance into a robot arm control policy.

Since artificial evolution of neural networks has been shown competitive and in many cases more efficient than other reinforcement learning methods (Moriarty and Miikkulainen 1996a; Whitley et al. 1993), the approach in this paper is based on neuro-evolution as the reinforcement learning method.

3 Evolving Neuro-Controllers

Recently there has been much interest in evolving neural networks with genetic algorithms in control tasks (Cliff et al. 1993; Moriarty and Miikkulainen 1996a; Nolfi et al. 1994; Whitley et al. 1993; Yamauchi and Beer 1993). Genetic algorithms (Holland 1975; Goldberg 1989) are global search techniques patterned after Darwin's theory of natural evolution. Numerous potential solutions are encoded in strings, called *chromosomes*, and evaluated in a specific task. Substrings, or *genes*, of the best solutions are then combined to form new solutions, which are inserted into the population. Each iteration of the genetic algorithm consists of solution evaluation and recombination and is called a *generation*. The idea is that structures that led to good solutions in previous generations can be combined to form even better solutions in subsequent generations.

Since genetic algorithms do not require explicit credit assignment to individual actions, they belong to the general class of reinforcement learning algorithms. In genetic algorithms, the only feedback that is required is a general measure of proficiency for each potential solution. Credit assignment for each action is made implicitly, since poor solutions generally choose poor individual actions. Thus, which individual actions are most responsible for a good/poor solution is irrelevant to the genetic algorithm, because by selecting against poor solutions, evolution will automatically select against poor actions.

In neuro-evolution, the solutions take the form of neural networks. Properties such as weights and connections are encoded in chromosomes, which are evolved by the ge-

netic algorithm. Neuro-evolution offers many important advantages to robotic arm control over supervised methods. First, it operates using only the overall performance of the neural network controllers as a guide. If avoiding obstacles is a necessary component of good performance, the genetic algorithm will select for networks that can avoid obstacles. No input/output examples are necessary, and thus neuro-evolution is not constrained by the inability to generate training examples. Second, neuro-evolution can be applied with very little *a priori* information. Knowledge of the robot arm dynamics, the arm's environment, or the components of the visual system is not necessary as they are in path-planning algorithms. Neuro-evolution evolves this knowledge through experience and tailors it's control policy to meet the specific demands of the domain.

4 A Neuro-Evolution Implementation for Robot Arm Control

This section describes a neuro-evolution algorithm and architecture for controlling the OSCAR-6 robot arm. Figure 1 shows a picture of an OSCAR robot. The current implementation is designed for evolution in a simulation model and then application to the real robot arm. Future work will study evolution directly with the OSCAR robot.

The network controller receives both camera-based visual and infrared sensory input representing the location of the target and the distance from obstacles, and must resolve a series of joint rotations to position the hand at a target location. The hardware specifications and algorithms used to generate the input are independent of the learning system and are considered given. For descriptions of camera-based vision and infrared sensors in robot manipulators, see (Lumelsky 1987; Papanikolopoulos and Khosla 1993; Sanderson and Weiss 1983; van der Smagt 1995; Wijesoma et al. 1993). The focus of this paper is how to automatically integrate the sensory information into an effective control policy.

4.1 Primary and Secondary Control Networks

The task of reaching a target can be seen as a composite of two basic movements. First, the robot arm must make several large joint rotations to get within a certain proximity of the target object. Such rotations often involve detecting and avoiding obstacles in the arm's path. Second, the robot arm must make smaller, more precise movements to position the end effector within grasping distance of the target object. This observation leads to an efficient design of a neuro-evolution system, where control is divided between two networks. The first, called the *primary network*, positions the arm near the target, while the *secondary network* makes the smaller movements to reach the target object.

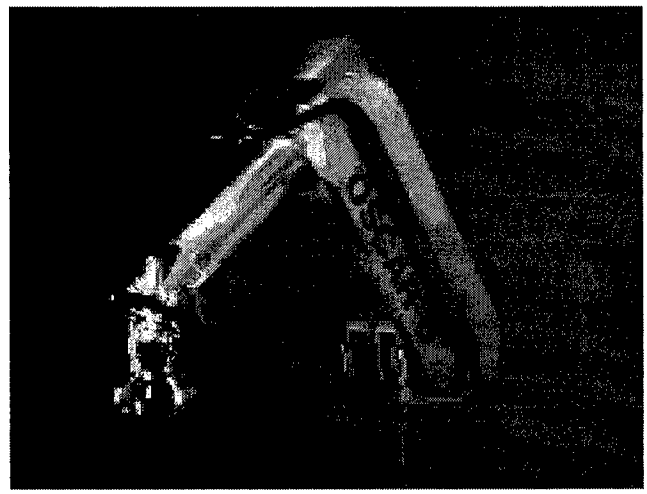


Figure 1: The OSCAR-6 robot arm. OSCAR is designed for pick-and-place tasks and has been applied in several industrial settings. The arm contains 6 total joints and a camera mounted in the end effector.

When a task is initiated, the primary network moves the arm until it specifies that the arm should be stopped or it exceeds a prespecified maximum number of joint rotations. There is no fixed proximity boundary for the primary network; its goal is to "get as close as it can". It is possible to evolve primary networks to complete the entire task, however, because of the diminishing returns of late evolution, it is often more advantageous to accept a certain level of proficiency and begin a new search for secondary networks. Since the secondary networks start relatively close to the target, there is normally little need for an obstacle avoidance strategy. Thus, any of the existing supervised approaches can generate effective secondary networks. Secondary networks are evolved in this paper to demonstrate that neuro-evolution can solve the entire task. Evolving the secondary networks also generates obstacle avoidance behavior for situations where the primary network leaves the arm very close to an obstacle.

4.2 Network Architectures

Each neural network controller (primary and secondary) contains 9 input, 16 hidden, and 7 output units (figure 2). The input units correspond to the x, y, and z relative distances of the hand to the target and six directional proximity sensors located on the hand that sense obstacles in the negative and positive x, y, and z directions. In other words, they sense obstacles in back of, in front of, to the left of, to the right of, above, and below the end effector. They have a 10 cm range and return the absolute distance to the obstacle. If no obstacle is currently within the sensor range, the activation is 10.0.

Each joint's rotation is determined by two unique output units. The first unit is linear and the sign of its total

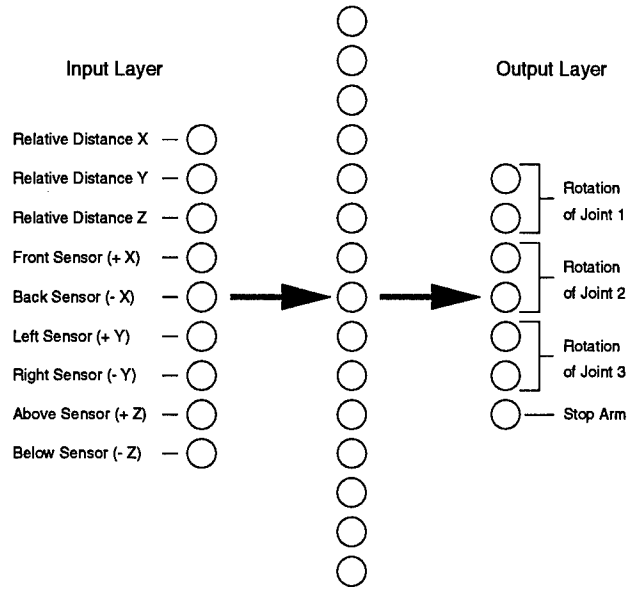


Figure 2: The architecture of the neural network controller for the OSCAR-6 robot. The dark arrows indicate activation propagation from the input to hidden layer and hidden layer to output layer. The connections and weights between the hidden layer and the input and output layers are evolved by the genetic algorithm.

activation specifies the direction of rotation. The second output unit is sigmoidal and specifies the magnitude of rotation. Dividing the output function this way makes it easier for hidden units to control a specific function, such as the direction of rotation of a particular joint. In the primary network, the magnitude output units are normalized between 0.0 and 5.0, limiting each joint rotation to [-5,+5] degrees. This forces the primary network to make several small joint rotations to reach the target, which allows it to more effectively sense and avoid obstacles in the arm's path. In the secondary network, the rotation is normalized between 0.0 and 1.0 to allow for fine movements near the target.

When the joint rotations are small, it is not necessary to take into account a large magnitude of distance from the target object. Such information can only interfere with the next local movement. Thus, it is very useful to "cap" the camera-based visual input units at 10.0 cm, such that if the target object is further away than 10.0 cm in any direction, the corresponding input unit receives an activation of only +/- 10.0.

A final threshold output unit is included as an override unit that can prevent movement regardless of the activations of the other output units. If the activation of the override unit is positive, the arm is not moved. If it is negative, the joint rotations are made based on the other output units. Without the override unit, stopping the arm would require setting the activation of the three sigmoidal units to exactly 0.0. Since genetic algorithms do not make systematic, small weight changes, it is very difficult to evolve neurons to compute such exact values. The override unit allows networks to easily stop the arm when it is sufficiently close to the target.

4.3 The Genetic Algorithm

The Hierarchical SANE (Symbiotic, Adaptive Neuro-Evolution) system was used to form the hidden layer connections in the neuro-control networks. SANE[1] was designed as a fast, efficient genetic algorithm for building neural networks in domains where it is not possible to generate training data for normal supervised learning. Symbiotic evolution has been evaluated in several tasks including the standard pole-balancing benchmark where it outperformed existing neuro-evolution and reinforcement learning approaches (Moriarty and Miikkulainen 1996a).

In contrast to standard neuro-evolution algorithms that evolve a population of neural networks, in SANE two separate populations are evolved: a *population of neurons* and a *population of network blueprints*. The neuron population provides efficient evaluation of the genetic building blocks, while the population of blueprints learns effective combinations of these building blocks.

Each individual in the blueprint population consists of a set of pointers to individuals in the neuron population. During each generation, networks are constructed by combining the hidden neurons specified in the blueprints. Each blueprint receives a fitness according to how well the corresponding network performs in the task. Each neuron receives a fitness according to how well the top five networks in which it participates perform in the task. A very aggressive genetic selection and recombination strategy is used to quickly build new structures in both the neuron and blueprint populations (see (Moriarty and Miikkulainen 1996b) for details).

SANE offers two important advantages over other neuro-evolution approaches. First, it maintains diverse populations. Because several different types of neurons are necessary to build an effective neural network, there is inherent evolutionary pressure to form neurons that perform different functions. Diversity allows recombination operators (crossover) to continue to generate new neural structures even in prolonged evolution, which ensures that the solution space will be explored efficiently. Second, SANE decomposes the search for solutions into a search for partial solutions. Instead of searching for complete networks all at once, solutions to smaller problems (good neurons) are evolved, which can be combined to form an effective full solution (a network).

[1]For the remainder of the paper, the name SANE will refer to the hierarchical version described in (Moriarty and Miikkulainen 1996b)

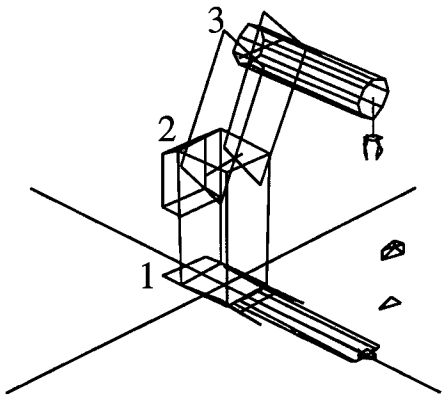


Figure 3: The Simderella simulation of the OSCAR robot. The numbers indicate the joints which are to be controlled.

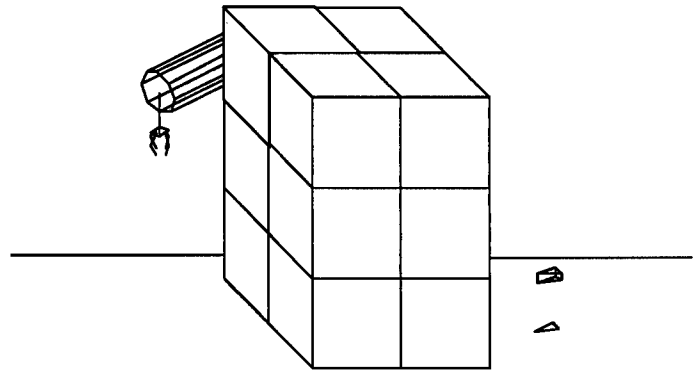


Figure 4: The 12 obstacle placements in the robot simulator. Each box is 30 × 30 × 30 centimeters. During any trial, one box is filled with an obstacle.

5 Evaluation

5.1 *Experimental Setup*

The Simderella 2.0 package written by van der Smagt (1994) was used as the robot arm simulator in these experiments. Simderella is a simulation of the OSCAR-6 anthromoporphic arm, and van der Smagt (1995) has reported that controllers that perform well in Simderella exhibit very similar performance when applied to OSCAR. Figure 3 illustrates the Simderella robot. Our eventual goal is to demonstrate evolved neuro-controllers in a real robot arm, however, currently simulation models are sufficient to test the viability of our approach.

Obstacles were introduced in the Simderella environment in the form of "boxes". Figure 4 shows the twelve different obstacle placements used in the simulations. During each trial, which consists of a sequence of moves to reach a target, one of the boxes is occupied by an obstacle. If the end effector moves into an occupied box, the trial ends, and the last position before the collision is used as the final position for fitness evaluation. This obstacle scheme is fairly primitive, since obstacles always have the same size and there is no check if the rest of the arm (besides the hand) violates an occupied box. However, the task is still quite difficult and, to our knowledge, no existing supervised learning approach can learn the intermediate joint rotations without significant *a priori* information about the size, shape, and location of the obstacles.

Each neural network evaluation begins with random, but legal, joint positions and a random target position. The target and hand are never started within an obstacle. The same reach space as van der Smagt (1995) is employed, where targets are placed within a 180 degree rotation of the first joint. A total of 450 target positions were created and separated into a 400 position training set and a 50 position test set. During evolution, a target position is randomly selected from the training set before each network evaluation. During each trial, a network is allowed to move the arm until one of the following conditions occur:

1. The network stops the arm.
2. The network places the arm in an illegal position (e.g. hits the floor).
3. The network hits an obstacle.
4. The number of moves exceeds 40.

The score for each trial is computed as the percentage of distance that the arm covered from its initial starting point to the target position. For example, if the arm started 120 cm from the target and its final position is 20 cm from the target, the network receives a score of $(120 - 20)/120 = 0.83$. The percentage of distance covered, instead of the absolute final distance, provides a fairer comparison between a network that receives a close target and a network that receives a distant target. Each network is evaluated over a single randomly selected target.

A population of 1600 neurons and 200 network blueprints are evolved by SANE. The first stage of evolution consists of only primary networks. The population is evolved for 200 generations, and the best network of each generation is tested over the 50 target test set. The overall best network is then fixed as the primary network, and the secondary network evolution begins from a random population.

5.2 *Results*

Figure 5 shows the performance of the primary networks per generation averaged over 10 simulations. The graph plots the average final distance of the best neural network found at or before each generation. On average, a network capable of moving the arm within an average of 10 cm was found within the first 100 generations.

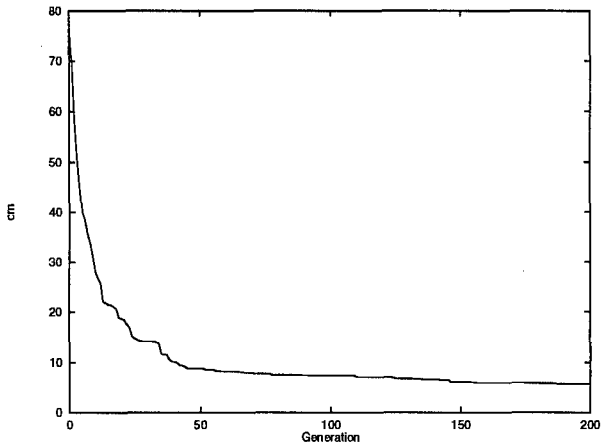


Figure 5: Evolution of primary networks. The average distance from the targets is plotted for the best network found so far at each generation. The curve is an average over 10 simulations; in each simulation, the distances were averaged over 50 randomly placed targets.

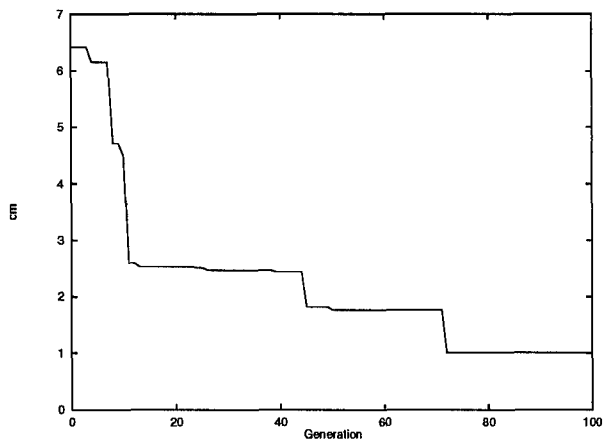


Figure 6: Evolution of secondary networks. The distance per generation is plotted, averaged over 50 trials and 10 simulations. In each simulation, a different primary network was used.

Figure 6 shows the performance of the secondary networks per generation. Again, the graph presents an average of 10 simulations. In each simulation, the primary network was taken from a different primary evolution. Within 80 generations, the secondary networks were able to position the arm with an error of only 1 cm, which is considered acceptable for most industrial applications (van der Smagt 1995). Thus, in this task, the combination of the primary and secondary networks can effectively control the robot arm to within industry standards.

It is difficult to measure how efficiently a network avoids obstacles as a function of each generation, since early networks do not hit many obstacles simply because they do not move the arm very far. Thus, counting the number of hits is a poor measure of obstacle avoidance. A better metric is the percentage of trials in which the

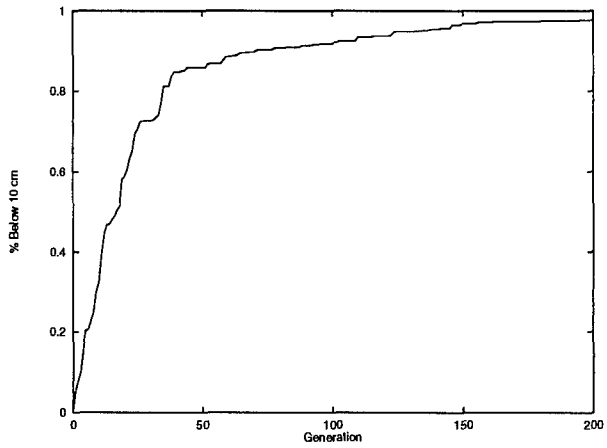


Figure 7: Evolution of obstacle avoidance behavior. The percentage of trials within 10 cm per generation is plotted for the primary network evolution.

primary network positions the arm within 10 cm of the target object. To achieve a high percentage, the network must contain a strong avoidance strategy coupled with an effective target reaching ability. Figure 7 plots this percentage for the best primary networks at each generation. On average, the best primary networks moved within 10 cm of the target objects 98% of the time. When collisions did occur, it was often not due to poor control decisions. With only 6 proximity sensors, blind spots are inevitable, and occasionally a nearby obstacle can not be detected. Thus given more sensors, the primary networks should encounter even fewer obstacles.

In order to discover how often it was necessary to avoid obstacles, a manually-designed inverse-kinematics controller[2] was tested on the 50 position test set with different box configurations. On average, the fixed controller, which takes the most direct path, hit obstacles in 11% of the trials. Since the best networks found in these simulations hit obstacles in only 2% of the trials, we can conclude that significant avoidance strategies have been evolved.

6 Future Work

The simple task described in this paper is already an important advancement in robot control, since it demonstrates that it is possible to learn a general obstacle avoidance strategy in a neuro-controller. However, the obstacle task needs to be scaled up before neuro-evolution can be claimed effective in real-world robot arm control. The approach will be scaled up in two stages: first, the obstacle task in the Simderella simulator will be extended to less regular obstacles and infrared sensors will be placed along the robot arm to prevent collisions with the entire arm. Second, the approach will be tested with a

[2]The controller is included as part of Simderella simulator

real robot arm. Simulation models are useful for evaluating new control algorithms, but important issues such as sensor noise and real-time control are often difficult to model. Thus, to truly validate our approach experiments are needed using real hardware.

Experiments will examine the feasibility of evolving neural network controllers directly with a real arm. Since domain models are expensive to generate and require significant *a priori* knowledge, it is important to study neuro-evolution as a model-free control system. While SANE does contain a very fast genetic search engine, the number of network evaluations currently required may expend too much time in real hardware. Improvements to the network architecture and fitness calculation may significantly reduce the number of evaluations. For example, dividing each joint movement between two output units (direction and magnitude) instead of one, resulted in about half as many evaluations to reach the same level of proficiency. In addition, much of our work will continue to focus on methods for improving the general efficiency of a neuro-evolution search.

The neural controller described in this paper is a fixed adaptive controller (Werbos 1992); once the controller is evolved it does not change. However, it would be desirable to build a controller that can adapt online to take advantage of domain specific information. Nolfi and Parisi (1995) have developed a method where evolved neural networks compute training signals for the the controller after every activation. Such networks could be used as online learning controllers by evolving the ability to adjust connection weights in response to the specific environment. Future work will study if evolved local learning can create more robust neuro-controllers and help networks evolved in simulation adapt to real hardware.

7 Conclusion

In many industrial settings, it is crucial for a robot arm to detect and avoid obstacles in the its path. Existing methods for learning robot arm control, however, cannot learn the intermediate joint rotations necessary to move around an obstacle. By evolving neuro-controllers with genetic algorithms, such rotations can be learned since performance is evaluated over multiple control steps. Experiments in a sophisticated simulation of the OSCAR-6 robot arm showed that neuro-evolution can effectively integrate both target reaching and obstacle avoidance into a single control policy. Future experiments will examine the application and scale-up potential of this approach to real robot arms.

Acknowledgments

The authors would like to thank Patrick van der Smagt for making his Simderella simulator available. This research was supported in part by the National Science Foundation under grant #IRI-9504317.

References

Baker, W. L., and Farrell, J. A. (1992). An introduction to connectionist learning control systems. In *Handbook of Intelligent Control*, 35–63. New York: Van Nostrand Reinhold.

Cliff, D., Harvey, I., and Husbands, P. (1993). Explorations in evolutionary robotics. *Adaptive Behavior*, 2:73–110.

Feddema, J. T., and Lee, G. C. S. (1990). Adaptive image feature prediction and control for visual tracking with a hand-eye coordinated camera. *IEEE Transactions on Systems, Man, and Cybernetics*, 20(5).

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning.* Reading, MA: Addison-Wesley.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence.* Ann Arbor, MI: University of Michigan Press.

Kawato, M. (1990). Computational schemes and neural network models for formation and control of multijoint arm trajectory. In *Neural Networks for Control.* Cambridge, MA: MIT Press.

Kuperstein, M. (1991). INFANT neural controller for adaptive sensory-motor coordination. *Neural Networks*, 4(2).

Lumelsky, V. J. (1987). Algorithmic and complexity issues of robot motion in an uncertain environment. *Journal of Complexity*, 3:146–182.

Miller, W. T. (1989). Real-time application of neural networks for sensor-based control of robots with vision. *IEEE Transactions on Systems, Man, and Cybernetics*, 19(4):825–831.

Moriarty, D. E., and Miikkulainen, R. (1996a). Efficient reinforcement learning through symbiotic evolution. *Machine Learning*, 22:11–32.

Moriarty, D. E., and Miikkulainen, R. (1996b). Hierarchical evolution of neural networks. Technical Report AI96-242, Department of Computer Science, The University of Texas at Austin.

Nolfi, S., Floreano, D., Miglino, O., and Mondada, F. (1994). How to evolve autonomous robots: Different approaches in evolutionary robotics. In *Artificial Life IV.* Cambridge, MA.

Nolfi, S., and Parisi, D. (1995). Learning to adapt to changing environments in evolving neural networks. Technical Report 95-15, Department of Neural Systems and Artificial Life, Institute of Psychology, CNR - Rome.

Papanikolopoulos, N. P., and Khosla, P. K. (1993). Adaptive robotic visual tracking: Theory and experiments. *IEEE Transactions on Automatic Control*, 38(3):429–444.

Sanderson, A. C., and Weiss, L. E. (1983). Adaptive visual servo control of robots. In Pugh, A., editor, *Robot Vision*, 107–116. New York: Springer-Verlag.

van der Smagt, P. (1994). Simderella: A robot simulator for neuro-controller design. *Neurocomputing*, 6(2).

van der Smagt, P. (1995). *Visual Robot Arm Guidance using Neural Networks*. PhD thesis, The University of Amsterdam, Amsterdam, The Netherlands.

Walter, J. A., Martinez, T. M., and Schulten, K. J. (1991). Industrial robot learns visuo-motor coordination by means of neural-gas network. In Kohonen, T., editor, *Artfificial Neural Networks*, vol. 1. Amsterdam.

Watkins, C. J. C. H., and Dayan, P. (1992). Q-learning. *Machine Learning*, 8(3):279–292.

Weiss, L. E., Sanderson, A. C., and Neumann, C. P. (1987). Dynamic sensor-based control of robots with visual feedback. *Journal of Robotics and Automation*, RA-3.

Werbos, P. J. (1992). Neurocontrol and supervised learning: An overview and evaluation. In *Handbook of Intelligent Control*, 65–89. New York: Van Nostrand Reinhold.

Whitley, D., Dominic, S., Das, R., and Anderson, C. W. (1993). Genetic reinforcement learning for neurocontrol problems. *Machine Learning*, 13:259–284.

Wijesoma, S. W., Wolfe, D. F. H., and Richards, R. J. (1993). Eye-to-hand coordination for vision-guided robot control applications. *The International Journal of Robotics Research*, 12(1):65–78.

Yamauchi, B. M., and Beer, R. D. (1993). Sequential behavior and learning in evolved dynamical neural networks. *Adaptive Behavior*, 2:219–246.

Automatic Generation of Adaptive Programs

Lee Spector*†
lspector@hampshire.edu

Kilian Stoffel †
stoffel@cs.umd.edu

*School of Cognitive Science and Cultural Studies
Hampshire College
Amherst, MA 01002

†Department of Computer Science
University of Maryland
College Park, MD 20742

Abstract

This paper shows how *ontogenetic programming*, an enhancement to the genetic programming methodology, allows for the automatic generation of adaptive programs. Programs produced by ontogenetic programming may include calls to self-modification operators. By permitting runtime program self-modification, these operators allow evolved programs to further adapt to their environments. In this paper the ontogenetic programming methodology is described and two examples of its use are presented, one for binary sequence prediction and the other for action selection in a virtual world. In both cases the inclusion of self-modification operators has a clear positive impact on the ability of genetic programming to produce successful programs.

1 Introduction

Genetic programming is a methodology for the automatic generation of computer programs by means of biologically-inspired processes of recombination and natural selection (Koza, 1992). Genetic programming systems process populations of computer programs. The initial populations typically consist of programs that are random combinations of elements from problem-specific function and terminal sets. The programs are assessed for fitness, usually by running them on representative sets of inputs, and the resulting fitness values are used in producing the next generation of programs. A variety of genetic operations, including fitness-proportionate reproduction, crossover, and mutation may be employed in constructing programs for the next generation. After a preestablished number of generations, or after the best fitness improves to some preestablished level, the best-of-run individual is designated as the result and is produced as the output from the genetic programming system.

Genetic programming systems are adaptive insofar as they support the adaptation of a population of programs to a particular fitness test throughout the course of evolution. In many problem environments it may also be advantageous for the evolved programs to *themselves* be adaptive. For example, it may be useful for evolved programs to perform on-line learning about complex environments, to adjust their own parameters as dynamic environments change over time, or even to adopt entirely new algorithms on the basis of environmental input. These sorts of runtime adaptive capabilities can be critical for achieving good performance in many real-world environments.

There are two obvious ways in which genetic programming systems could support runtime adaptation. The first involves the runtime manipulation of dynamic data structures upon which program execution may depend. Indexed memory (Teller, 1994) and memory terminals (Iba et al., 1995) both provide the required functionality; programs that use these mechanisms may acquire and store information from their environments at runtime, and they may use this information to guide future behavior.

A second, more direct mechanism is described in this paper: program self-modification operators are included in the set of functions that may be used by evolved programs. By use of these operators, evolved programs can dynamically change their own structure—and thereby their future behavior—during the course of a run. A program's self-modification strategy is itself evolved; it may be arbitrarily complex and it may be conditionalized on runtime environmental inputs.

Biologists refer to the developmental progression of an individual through its life span as *ontogeny*. The inclusion of program self-modification operators in a genetic programming function set allows for the evolutionary production of programs with rich ontogenetic components. For this reason, the technique is called *ontogenetic programming*, and the program self-modification operators are called *ontogenetic operators*.

Some other evolutionary computation frameworks already allow for runtime adaptation. This is generally

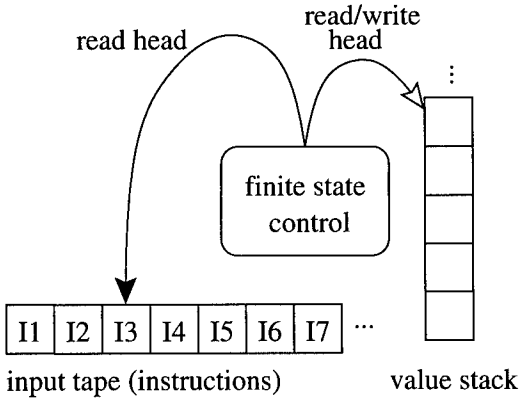


Figure 1: The HiGP virtual stack machine

accomplished by using an underlying adaptive mechanism that is only partially pre-configured by evolutionary processes; additional configuration and adaptation is accomplished by the underlying adaptive mechanism as it confronts its environment. For example, genetic algorithms have been used to evolve the initial parameter vectors for dynamical neural networks that integrate runtime sensor information to modify their own future behavior (Yamauchi and Beer, 1994). In these sorts of systems the runtime adaptive mechanisms (e.g., learning procedures for neural networks) are fixed; by contrast, ontogenetic programming allows for the simultaneous evolution of runtime adaptive mechanisms and the programs that use them.

The remainder of this paper demonstrates the automatic generation of adaptive programs using ontogenetic programming. The HiGP genetic programming system, which operates on linear programs for a stack-based virtual machine, is used for this demonstration. Program modification mechanisms for linear programs are particularly simple, but the technique is not limited to systems with linear programs; see (Spector and Stoffel, 1996) for a discussion of ontogenetic programming with more traditional S-expression-based representations. After the description of the technique two examples are provided: a simple but illustrative binary sequence prediction problem, and a more complex action selection problem. The paper concludes with a brief description of work yet to be done.

2 Ontogenetic HiGP

HiGP is a high-performance genetic programming engine that combines techniques from string-based genetic algorithms, S-expression-based genetic programming systems, and high-performance parallel computing systems (Stoffel and Spector, 1996). It is a fast, flexible, and portable system with an efficient parallel implementation that scales nearly linearly with the number of available processors. HiGP produces and manipulates linear programs for a stack-based virtual machine (as in (Perkis, 1994)), rather than the tree-structured S-expressions used in traditional genetic programming.

The HiGP virtual machine is similar to standard pushdown automata models (see, e.g., (Kain, 1972)) and to the execution model of the FORTH programming language (Brodie, 1981). It consists of an input tape containing a linear program to be executed, a pushdown stack for function arguments and return values, and a finite-state control unit that controls the execution of the program (see Figure 1). The program on the input tape must be composed of words that have been pre-defined as HiGP operators. To execute a program, the finite-state control unit simply reads the program from the input tape and executes the defined function call for each operator. Operators may be defined to perform arbitrary computations and to manipulate the values on the stack. They may also reposition the read head on the input tape, thereby allowing for conditional branch operators and loops. The stack is usually initially empty, although for some applications it may be pre-loaded with inputs for the program. Program execution halts when the last operator on the input tape has been processed, or when a pre-established maximum number of operators have been processed. Program results are read from the top of the stack at the end of program execution. Note that while HiGP evolves programs for the virtual machine, the machine itself never changes—it is simply the execution model for the evolving programs, much as a Lisp-like S-expression evaluator is the execution model for standard genetic programming (Koza, 1992). Further details about HiGP and the HiGP virtual machine can be found in (Stoffel and Spector, 1996).

One normally provides a family of `push` operators that correspond to the terminal set in a traditional genetic programming system; each `push` operator pushes a pre-determined value onto the stack. Because all programs in the system have the same length, and because we do not wish to pre-determine the number of actual problem-solving operators that should appear in solution programs, a `noop` operator is also usually included. With the inclusion of the `noop` operator, which does nothing, the fixed program size becomes a size *limit*, analogous to the depth limits used in S-expression-based genetic programming systems. In other words, "shorter" programs can be encoded by filling in extra program steps with `noops`.

Additional operators may be easily added to the system. The only restrictions are that they must take their arguments from the stack and that they must push their results back onto the stack. When there are not enough values on the stack to serve as arguments for an opera-

tor it is skipped by the finite-state control unit and the stack remains untouched (as in (Perkis, 1994)).

A simple example may help to clarify the operation of the virtual stack machine. Consider the following program:

```
push-x noop push-y * push-x push-z noop - +
noop noop
```

The noops in the program have no effect and the remainder is equivalent to the C expression:

```
(x * y) + (x - z)
```

The ontogenetic version of HiGP results from adding the following program self-modification operators to the function set:

segment-copy copies a part of the program over another part of the program. The function takes three arguments from the stack, all of which are coerced to integers in the appropriate ranges: the start position of the segment to copy (relative to the current instruction), the length of the segment, and the position to which the segment should be copied (relative to the current instruction). If there are not three values on the stack the instruction is skipped.

shift-left rotates the program to the left. The call takes one argument from the stack, which is coerced to a positive integer if necessary: the distance by which the program is to be rotated. If there is no value on the stack the program is rotated by one instruction to the left.

shift-right rotates the program to the right. The call takes one argument from the stack, which is coerced to a positive integer if necessary: the distance by which the program is to be rotated. If there is no value on the stack the program is rotated by one instruction to the right.

The position of the current instruction pointer is not changed by the execution of an ontogenetic operator. For example, if instruction #23 is a `segment-copy`, then after its execution instruction #24 will be executed, regardless of the fact that the *old* instructions #23 and #24 may now have been moved elsewhere.

3 Example 1: Binary Sequence Prediction

The ontogenetic version of HiGP can solve problems that cannot be solved by the ordinary version of HiGP. This section demonstrates the application of ontogenetic programming to a binary sequence prediction problem for which this is the case. Because indexed memory also provides a runtime adaptive capability, one might conjecture that the use of indexed memory would be sufficient for the solution of this problem. But this section shows that this is not the case; the addition of indexed memory to the ordinary version of HiGP is not sufficient to produce solutions to the described problem. In other words, the ontogenetic extensions provide benefits that indexed memory does not.

Consider the *sequential regression* problem, a variant of the symbolic regression problem (Koza, 1992). As in ordinary symbolic regression, the goal in sequential regression is to produce a program that returns the appropriate y value for each x value in a data set. In the sequential regression problem it is additionally stipulated that the x values will be presented in a particular order. Programs are run multiple times, one for each x value in the fitness-testing range, and they always encounter these x values in the same order. Runtime adaptation during the course of a program's progression through the x range may be useful for sequential regression problems because different programs may be most appropriate for different ranges of the target function. Further, transitions from one value to the next may be better mediated by ontogenetic operators than by domain operators.

Consider a binary sequential regression problem with a target function that repeats the values [0 1 0 0 0 1] as x increases. We will call this problem the *binary sequence prediction problem.* The goal is to produce a program that returns the appropriate y value from the sequence for each x (index) value; the program will be run once for each x value, and the x values will be presented in order from 0 to some number n. Several related sequence prediction problems have been described in the literature. In particular, Iba et al. describe a related binary oscillation task (Iba et al., 1995).

For our experiments we assessed fitness by testing each program on the range [0–17]. Positive return values were mapped to 1, and negative return values were mapped to 0; it was therefore sufficient for a program to return either 0 or any negative number in place of each 0, and any positive number in place of each 1. Fitness was calculated as the number of incorrect answers; lower fitness values therefore indicated better programs, and a fitness value of 0 indicated a completely correct program. We used a function set consisting of the 2-argument addition function +, the 2-argument subtraction function -, the 2-argument multiplication function *, the 2-argument protected division function % (Koza, 1992), and the 0-argument `push-x` function for the independent variable x.[1] In addition, we included the stack

[1] In previous experiments we also included `push-0` and `push-1` in the function set (Spector and Stoffel, 1996). Surprisingly, their removal had almost no effect on the results.

manipulation function dup, which pushes a duplicate of the top element onto the stack, and noop. A copy of x was pre-loaded onto the stack prior to each program execution.

100 runs of the ordinary (non-ontogenetic) version of HiGP were conducted on this problem with a population size of 100, a maximum program size of 30, a crossover rate of 90%, a reproduction rate of 10%, and a maximum of 20 generations per run. No correct solutions were produced by any of the 100 runs. One can conclude that the problem cannot reliably be solved by ordinary HiGP and the given parameters. Small numbers of runs were also conducted with varied parameters (for example, with mutation and with larger populations), but no correct solutions were ever produced.

100 runs of the ontogenetic version of HiGP were then conducted on this problem (adding the `segment-copy`, `shift-right` and `shift-left` functions to the function set). The population size and other parameters were the same as those described above. 12 completely correct solutions were produced by the 100 runs. 10 of these solutions appeared to be general; although fitness was assessed only over the range [0–17], these programs produced correct results over the range [0–39], and they appeared to be correct to any limit. The following is a correct evolved program:

```
+ push-x - noop - shift-left shift-right
push-x segment-copy shift-left shift-right
* + segment-copy + % * * noop noop
shift-right + % shift-left shift-right
noop noop - noop *
```

Because the ontogenetic operators modify programs *as they run*, it is difficult to trace program execution or to get an intuitive feel for program self-transformation strategies. Nonetheless, it is sometimes interesting to look at some of forms through which a program passes. The above program looks as follows after the completion of 9 full executions:

```
- shift-left + % * * noop noop shift-right %
shift-left noop noop - noop * + push-x - noop
- shift-left + % * * noop noop shift-right %
```

At the end of 18 executions it appears more similar to, but still different from, its initial state:

```
+ push-x - noop - shift-left shift-right
segment-copy shift-left * + segment-copy + %
* * noop noop shift-right % shift-left noop
noop - noop * + push-x - noop
```

Although equivalent programs do recur through the 18 executions of the fitness-test range, no obvious alignment of these recurrences to the 6-element sequence value pattern is evident.

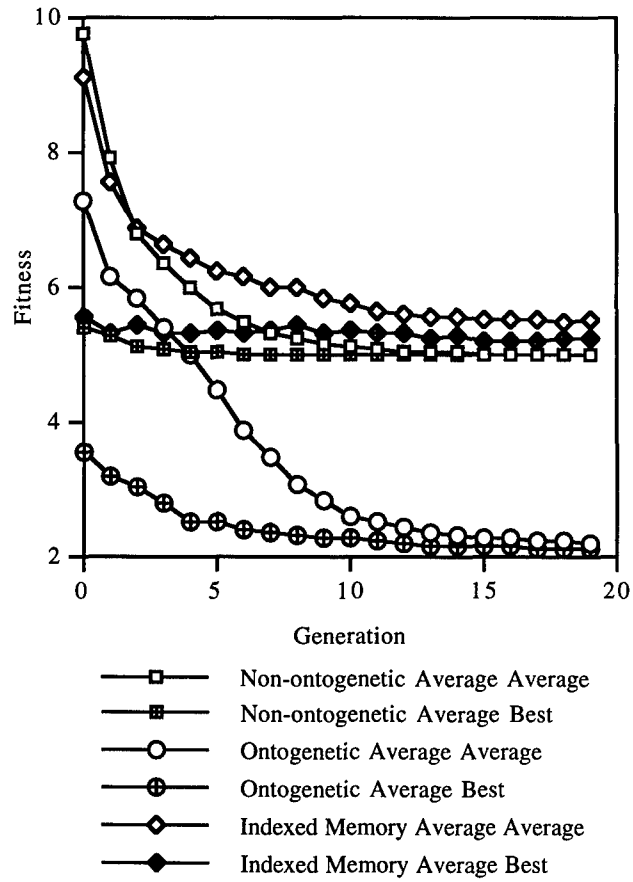


Figure 2: Average and best fitnesses by generation, averaged over the 100 runs for each condition, for the binary sequence prediction problem.

Because indexed memory also provides a limited runtime adaptive capability, 100 runs of HiGP with indexed memory were also conducted on this problem. A 30-element indexed memory was added, with a 2-argument `write` function that stores the value of one argument in the location indexed by the other, and a 1-argument `read` function that pushes the indexed element of the memory onto the stack. The population size and other parameters were the same as those described above. No correct solutions were produced by any of the 100 runs. One can conclude that indexed memory does not provide sufficient runtime adaptive power for HiGP to reliably solve this problem with the given parameters.

Figure 2 shows graphs of average and best fitnesses by generation, averaged over the 100 runs conducted for each condition (non-ontogenetic, ontogenetic, and indexed memory). From this graph it is clear that the advantage provided by the ontogenetic operators is indeed significant. Note that in all conditions the average average fitness appears to converge to a value close to

the average best fitness. Note also that indexed memory actually makes matters worse; the system performs better without it.

4 Example 2: Action Selection in Wumpus World

Wumpus world (Russell and Norvig, 1995) is an environment in which an agent must select actions to avoid death and to achieve goals. The use of genetic programming for the evolution of Wumpus world agents has been previously described (Spector, 1996). In this section we briefly describe the Wumpus world problem and the results of experiments that use ontogenetic programming to produce Wumpus world agents.

Wumpus world is cave represented as a grid of squares surrounded by walls. The agent's task is to start in a particular square, to move through the world to find and to pick up a piece of gold, to return to the start square, and to climb out of the cave. The cave is also inhabited by a "wumpus"—a beast that will eat anyone who enters its square. The wumpus produces a stench that can be perceived by the agent from adjacent (but not diagonal) squares. The agent has a single arrow that can be used to kill the wumpus. When hit by the arrow the wumpus screams; this can be heard anywhere in the cave. The wumpus still produces a stench when dead, but it is harmless. The cave also contains bottomless pits that will trap unwary agents. Pits produce breezes that can be felt in adjacent (but not diagonal) squares. The agent perceives a bump when it walks into a wall, and a glitter when it is in the same square as the gold.

The wumpus world agent can perform only the following actions in the world: go forward one square; turn left 90°; turn right 90°; grab an object (e.g., the gold) if it is in the same square as the agent; release a grabbed object; shoot the arrow in the direction in which the agent is facing; climb out of the cave if the agent is in the start square.

The agent's program is invoked to select a single action for each time-step of the simulation. The program returns one of the valid actions and the simulator then causes that action, and any secondary effects, to happen in the world. The agent can maintain information between actions by use of a persistent memory system. The agent's program has a single parameter, a "percept" that encodes all of the sensory information available to the agent. The agent's program can refer to the components of the percept arbitrarily many times during its execution.

Agents are assessed on the basis of performance in four worlds.[2] In each world the agent is allowed to perform a maximum of 50 actions, and the agent's score is determined as follows: 100 points are awarded for obtaining the gold, there is a 1-point penalty for each action taken, there is a 100-point penalty for getting killed, and there is a 100-point penalty for each unit of distance between the agent and the gold at the end of the run. Agents are not explicitly rewarded for climbing out of the cave, although less action penalties are accumulated if an agent climbs out and thereby ends the simulation. An agent is considered to have solved the problem if its average score in four worlds is greater than zero. To have obtained such a score an agent must

Table 1: Wumpus World Operators

Name	Args	Description
+	2	Pushes the sum of the arguments modulo 7 onto the stack.
-	2	Pushes the difference of the arguments modulo 7.
*	2	Pushes the product of the arguments modulo 7.
and	2	Pushes 1 if both arguments are non-zero, or 0 otherwise.
or	2	Pushes 1 if either or both arguments are non-zero, or 0 otherwise.
not	1	Pushes 1 if the argument is 0, or 0 otherwise.
ifz	3	"If zero"—jumps forward the number of instructions specified in the second argument if the first argument is 0; jumps forward the number of instructions specified in the third argument otherwise.
read	2	Pushes the contents of the 2-dimensional indexed memory location indexed by the arguments. Memory locations contain 0 if they have not yet been written to.
write	3	Pushes the *previous* contents of the memory location indexed by the first two arguments, and then fills the memory location with the third argument.
noop	0	Does nothing.
{0 - 6}	0	Each pushes the corresponding number.
rand7	0	Pushes a random integer between zero and six (inclusive).
stench	0	Pushes 1 if the current percept includes a stench (from the wumpus), or 0 otherwise.
breeze	0	Pushes 1 if the current percept includes a breeze (from a pit), or 0 otherwise.
glitter	0	Pushes 1 if the current percept includes a glitter (from the gold), or 0 otherwise.
bump	0	Pushes 1 if the current percept includes a bump (from a wall), or 0 otherwise.
sound	0	Pushes 1 if the current percept includes a sound (from the wumpus), or 0 otherwise.

[2] In (Spector, 1996) four new random worlds were generated for each fitness test. This allowed very simple programs that were fortunate enough to be tested on appropriately simple worlds to appear to be quite successful. In the experiments described in this paper four random worlds were generated for each run of the genetic programming system, and the same four worlds were used for each fitness test. This appears to make the problem considerably more difficult, although it is possible that the resulting agents will be overfitted to the particular four worlds and therefore less robust than some of those produced by the previous technique.

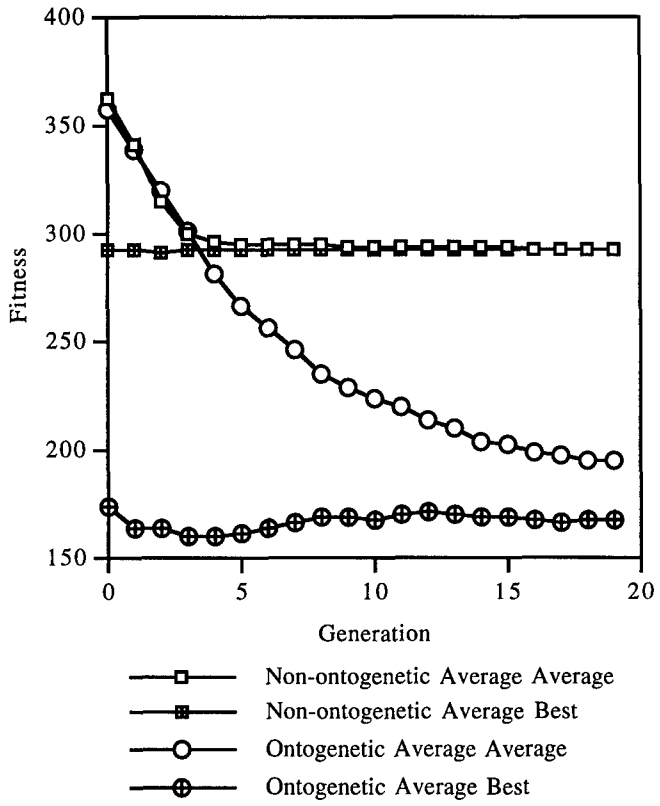


Figure 3: Average and best fitnesses by generation, averaged over the 200 runs for each condition, for the Wumpus world problem.

have grabbed the gold in at least one and usually two or more of the worlds, and it can have died in at most one of the four random worlds. This is difficult; in many cases it is necessary to risk death in order to navigate to the gold, and in some cases the gold may be unobtainable because it is in a pit or in a square surrounded by pits. "Standardized fitness" values (for which lower values are better (Koza, 1992)) are the average of the scores from the four worlds, subtracted from 100.

For the experiments reported here, the seven valid actions were mapped onto the integers from zero to six (inclusive), and a simple function set was used that consists primarily of simple arithmetic operators that manipulate and return numbers in this range.[3] The agent's memory system was implemented as a two-dimensional (7 by 7) indexed memory (Teller, 1994), each element of which could hold a single number. The complete function set is shown in Table 1.[4] For the ontogenetic runs, the above-described `shift-left`, `shift-right`, and `segment-copy` operators were also included. A program's result was computed as the absolute value, modulo 7, of the top element on the stack at the end of program execution.

200 runs of the ordinary (non-ontogenetic) version of HiGP were conducted on this problem with a population size of 200, a maximum program size of 100, a crossover rate of 89.5%, a reproduction rate of 10%, a mutation rate of 0.5%, tournament selection with a tournament size of 3, and a maximum of 20 generations per run. No correct solutions were produced by any of the 200 runs. One can conclude that the problem cannot reliably be solved by ordinary HiGP and the given parameters. Note that the Wumpus world problem always includes an indexed memory, and thereby supports some forms of runtime adaptation even without the use of ontogenetic operators.

200 runs of the ontogenetic version of HiGP were then run on this problem (adding the `segment-copy`, `shift-right` and `shift-left` functions to the function set). The population size and other parameters were the same as those described above. 10 solutions were produced by the 200 runs. The following is one of the evolved programs:

```
noop and 3 write - - not + and + 3 2 noop 6 *
write 5 4 1 ifz 1 + + 6 read 1 and + + 2
shift-right 1 stench breeze + or * 0 breeze +
or 1 2 4 shift-left 3 bump not 1 ifz 0 1 6
read glitter 5 segment-copy not 3 shift-left
shift-right write write * stench - 6 bump
sound - 6 noop bump glitter 0 3 - bump 0 0
sound bump stench 4 * or 1 ifz and 5 2 bump 5
* 5 write 6 and 1 -
```

Figure 3 shows graphs of average and best fitnesses by generation, averaged over the 200 runs conducted for each condition (non-ontogenetic and ontogenetic). This graph shows that with the ontogenetic operators the lowest average best fitness occurs early, around generation 4. Without the ontogenetic operators the average best fitness is considerably worse, it never improves, and it is nearly matched by the average average fitness—that is, the populations appear to have converged on non-solutions. It is therefore unlikely that longer runs (over more generations) would produce solutions without ontogenetic operators. It is clear from the graph that a significant advantage is provided by the ontogenetic operators.

5 Conclusions and Future Work

Ontogenetic programming, an enhancement to the genetic programming methodology, allows for the auto-

[3]The mapping from integers to actions was: 0 = forward, 1 = turn right, 2 = turn left, 3 = shoot, 4 = grab, 5 = release, 6 = climb.

[4]This function set, developed for HiGP, differs slightly from that used in (Spector, 1996), which used S-expression-based genetic programming.

matic generation of programs that adapt to their environments at runtime through the use of program self-modification operators. The ontogenetic programming methodology was described and applied to two examples: binary sequence prediction and action-selection in a virtual world. Runtime adaptation turns out to be useful for both of these problems, and the availability of ontogenetic operators allowed for the evolution of solutions in cases for which ordinary genetic programming failed.

Much additional work must be completed to assess the real value of this technique. Neither of the problems described in this paper are real-world problems, and it is not yet clear that the technique will scale well. It is not even entirely clear what features of an environment are related to the general utility of runtime adaptation. One speculation is that unpredictable dynamism in an environment favors runtime adaptation, but one can make the case that neither of the environments described in this paper have this feature. A program that solves the binary sequence prediction problem must only master a single fixed sequence. Although each program *predicts* the sequence dynamically, one element at a time, the sequence itself does not change throughout a run. Similarly, in the Wumpus world experiments described here, each agent had only to contend with a fixed sequence of four essentially static worlds. But for both of these problems runtime adaptation was nonetheless sufficiently useful to allow for solutions in cases in which no solutions could otherwise be produced. Additional applications of ontogenetic programming to agents in more complex and dynamic real-world environments will help to resolve some of these issues.

More work should also be conducted to examine the self-modification strategies actually employed by successful programs. Many possibilities exist; for example, programs might be copying segments of code into multiple program locations, thereby reusing code modules and obtaining an effect similar to that obtained with automatically defined functions (Koza, 1994) or automatically defined macros (Spector, 1996). It would also be interesting to trace the patterns of recurrence of particular program configurations as a program runs.

The ontogenetic programming technique can be varied in many ways—for example, by changing the set of provided ontogenetic operators. No systematic study of such variations has yet been performed. The use of ontogenetic programming with traditional S-expression-based genetic programming systems has been described (Spector and Stoffel, 1996), but the impact of program representation on the utility of particular ontogenetic operators has not yet been studied.

Considering the importance of adaptation in successful complex systems, it seems reasonable to conjecture that results presented in this paper will generalize in some manner—that genetic programming systems that adaptively generate *adaptive* programs will generally be more useful than those that do not. Ontogenetic programming provides the required capability, although more research must be conducted on the efficient exploitation of this capability.

More broadly, ontogenetic programming systems may present new opportunities for the general study of interactions between evolutionary and developmental processes. Biological systems are adaptive at both evolutionary (phylogenetic) and developmental (ontogenetic) levels; computational models of such systems should be adaptive at both levels as well.

Acknowledgments

Mark Feinstein helped to develop our initial interest in ontogeny, and to refine our understanding of its role in biological systems. Discussions at the 1995 AAAI Fall Symposium on Genetic Programming (Siegel, 1995) helped to further refine our approach to ontogenetic programming. The comments of two anonymous reviewers lead to several improvements in this paper. This research was supported in part by grants from ONR (N00014-J-91-1451), AFOSR (F49620-93-1-0065), and ARPA contract DAST-95-C0037.

References

Brodie, L. 1981. *Starting FORTH.* Prentice Hall.

Iba, H., T. Sato, and H. de Garis. 1993. Temporal Data Processing Using Genetic Programming. In *Proceedings of the 6th International Conference on Genetic Algorithms, ICGA-95*, edited by Larry J. Eshelman, pp. 279–286. San Francisco: Morgan Kaufmann Publishers, Inc.

Kain, R.Y. 1972. *Automata Theory: Machines and Languages.* New York: McGraw-Hill Book Company.

Koza, J.R. 1992. *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* Cambridge, MA: The MIT Press.

Koza, J.R. 1994. *Genetic Programming II: Automatic Discovery of Reusable Programs.* Cambridge, MA: The MIT Press.

Perkis, T. 1994. Stack-Based Genetic Programming. In *Proceedings of the 1994 IEEE World Congress on Computational Intelligence*, pp. 148–153. IEEE Press.

Russell, S.J., and P. Norvig. 1995. *Artificial Intelligence, A Modern Approach.* Englewood Cliffs, NJ: Prentice Hall.

Siegel, E.V., editor. 1995. *Collective Brainstorming at the AAAI Symposium on Genetic Programming*, http://www.cs.columbia.edu/~evs/gpsym95.html.

Spector, L. 1996. Simultaneous Evolution of Programs and their Control Structures. In *Advances in Genetic Programming 2*, edited by P.J. Angeline and K.E. Kinnear, Jr., pp. 137–154. Cambridge, MA: The MIT Press.

Spector, L., and K. Stoffel. 1996. Ontogenetic Programming. In *Proceedings of the Genetic Programming 1996 Conference.* Cambridge, MA: The MIT Press. In press.

Stoffel, K. and L. Spector. 1996. High-Performance Genetic Programming. In *Proceedings of the Genetic Programming 1996 Conference.* Cambridge, MA: The MIT Press. In press.

Teller, A. 1994. The Evolution of Mental Models. In *Advances in Genetic Programming,* edited by Kenneth E. Kinnear, Jr., pp. 199–219. Cambridge, MA: The MIT Press.

Yamauchi, B. and R. Beer. 1994. Integrating Reactive, Sequential, and Learning Behavior using Dynamical Neural Networks. In *From Animals to Animats 3*, edited by D. Cliff, P. Husbands, J. Meyer, and S.W. Wilson, pp. 382–391. Cambridge, MA: The MIT Press.

Sexual Swimmers

Emergent Morphology and Locomotion Without a Fitness Function

Jeffrey Ventrella
ventrell@media.mit.edu
Rocket Science Games, Inc.
139 Townsend St. San Francisco, CA 94107

Abstract

A virtual ecosystem is described which demonstrates emergent morphology and locomotion among a population of 2D figures in a virtual pond. A physics model enables a large variety of articulated figures to potentially propel themselves through simulated water. Swimmers with better skills at locomotion and turning are able to eat more food to gain energy and mate with other swimmers, thereby supplying more "fit" genetic building blocks to future generations. There is no fitness function used in this genetic algorithm scheme: fitness equals reproduction. Sexual selection is also modeled: genetically inherited preferences for mate color affect the distribution of phenotypes across the pond: swimmers of differing colors infrequently mate with each other, often breaking the population into distinct coloration groups. To witness the emergence of behavior on local scales (swimming styles), intermediate scales (family dramas), and global scales (population dynamics), this simulation can be viewed with an interactive "microscope" for panning and zooming. This work demonstrates how local behavior can affect (and be affected by) global behavior, and how these different levels of behavior evolve together.

1 Introduction

The majority of species on earth have evolved skills for locomotion in water, and the variation in morphology among species is matched by a great variety of swimming styles. For most species, locomotion is necessary for getting to food and, for many, to avoid becoming someone else's food. Locomotion also plays a role in the reproductive lives of most species. In order for an individual to mate with another of its species, it needs to approach this individual to be close enough—obviously—to mate. The simulation described in this paper bases its ability to generate emergent locomotion and morphology from these two simple elements of nature—food and sex.

This paper describes a simulation which drives an artificial life movie—a movie with no script. The actors are a variety of 2D figures, called *swimmers*, which inhabit a virtual pond. The swimmers' bodies consist of interconnected line segments of many colors, with movable joints. The motions of body parts can exert forces against the water and thus potentially accelerate a swimmer through the water. In this world, swimmers eat, reproduce, and die. Swimmers who are able to swim to food can replenish their energy. Those who cannot, die of starvation. Swimmers who are both able to eat and able to swim to a chosen mate, reproduce. Those who cannot mate, do not pass on their genes to future swimmers before dying. Evolution of improved swimming results. On the local scale (individual swimming styles) as well as the global scale (group population dynamics), emergent phenomena can be observed.

Sexual selection is modeled here as well—each swimmer has a genetically inherited favorite color in potential mates, which influences which swimmer it chooses from those within its view. These preferences are randomized at initialization, along with all the other genes. Eventually, in maturing populations, swimmers' color preferences begin to correspond with their actual body colors, often catalyzing distinct groups to emerge, in which case, the population becomes sympatric, with groups rarely interbreeding, particularly among the "purest" races, in which body color is homogeneous.

The use of color in modeling mate preference is not for aesthetics (not *ours*, at least—perhaps the swimmers'), but as a readily-visualized way of studying the effects of mate preference.

1.1 Autonomous Reproduction

Operators derived from the genetic algorithm (GA) are used in this simulation, but not in the conventional manner: in this case there are no discrete *generations*, and there is no explicit use of a fitness function for assigning fitness values to individuals in a population, thereby determining the rate at which they can reproduce. Instead, creatures who are able to swim towards a desired mate *automatically* reproduce, by virtue of the fact that they are able to swim to their goal. The fitness landscape is continually changing, as in the natural world. Fitness, then, is equivalent to reproduction, which, in this world, is equivalent to coming in contact with a desired mate.

At no time is any "outside assistance" used in the simulation to help encourage the emergence of optimized swimming—other than in the initial distribution of swimmers and food, and a set of swimmer perception and behavior settings. There is nothing but the situation at hand to determine how the population will evolve. In many simulation runs, the entire population of swimmers dies off before any significant results occur. This is the occasional price to pay for a decidedly *hands off* approach.

This approach also offers no means of exploiting a good solution when it happens somewhere in the pond. An Olympic individual can emerge within the pond, but if it has an inappropriate preference for mates, is not desired by other mates, or is born in a low swimmer or food populated area, it could die without ever having reproduced. This simulation runs on a mixture of skill and luck.

This work can be seen as an attempt to extend recent methods in generating emergent morphology and motor control in virtual creatures using GA's. In this simulation, the changing environmental conditions (and not explicitly a "Creator's" objective function) determine emerging behaviors, which are transformed by (and likewise transform) the environment. The emphasis, then, is not on simply optimizing behavior within a population, but in modeling and studying the heterogeneous outcome of a population of creatures living their lives within an ever-changing, varying ecosystem.

1.2 Real-time Animation

The simulation described here has an interactive component—the design of which has been a useful investment for development and analysis. One can view the running of a simulation using an interactive "microscope" which can zoom in and out, or pan across the pond. With this microscope, different parts of the simulation can be studied—one can watch a single swimmer up close; view a genetically related group of swimmers in one area of the pond; or view the entire pond to witness large-scale dynamics.

Populations of about 200 or less can be watched at real-time computer animation rates (10 or more frames per second) on an SGI Indigo2 computer. In experiments, the population count is usually initialized to 1000—which is too high for real-time animation: it is not yet visually informative. But the population inevitably drops way below this number as the ecosystem stabilizes, leaving a smaller number of statistically better swimmers. After this, informative and enjoyable animations can be watched for hours.

2 Related Work

The blossoming field of *Artificial Life* (Langton, 89) includes contributions from many disciplines. Computer animation techniques which complement traditional animated scripting with autonomous agents have made possible complex life-like systems composed of many distributed elements (Reynolds, 87). Physically-based modeling techniques and virtual motor control systems inspired by real animals are used to automate many of the subtle, hard-to-design nuances of animal motion (Badler, 91). In *task-level* animation, (Zeltzer, 91), and the *space-time constraints* paradigm, (Witkin and Kass, 88), these techniques allow an animator to direct an autonomous agent on a higher level.

Genetic algorithms (Holland, 75), (Goldberg, 89) have been applied towards artificial evolution of goal-directed motion in physically-based animated figures (Ventrella, 90). These include techniques for evolving stimulus-response mechanisms for locomotion (Ngo and Marks, 93), and for morphological variation in 3D forms (Sims, 94-1, 94-2), (Ventrella, 94). These techniques have been developed most comprehensively in the virtual creatures of Sims, through genetic programming (Koza, 92), and include evolution of locomotion in viscous fluids. An extensive model of fish locomotion, with perception and learning has been developed by Terzopoulos (94), and generates beautifully realistic animations.

The modeling of adaptive organisms imbedded in artificial ecosystems takes genetic algorithms a step further towards a Darwinian definition of fitness by allowing reproduction to occur spontaneously (Ray, 91). "Electronic primordial soups" involving spatiality, such as Yaeger's Polyworld (94), demonstrate artificial ecosystems in which mating, eating, learning, and even social behaviors, evolve within the simulated world.

Todd and Miller (91) have demonstrated how assortative sexual selection can drive a population to have arbitrary phenotypic features, above and beyond the features resulting from natural selection. They demonstrated how mate preference mechanisms could result in sympatric populations—breaking off into distinct non-interbreeding groups.

3 The Swimmer World

Swimmers is an extension of some previous projects by the author, involving stick figures whose motions are controlled by a set of parameters which evolve with the help of a GA. This project takes a step down in dimension (from 3 to 2), but includes a more automated means of evolving locomotion, by imbedding the figures in a dynamic environment, where the fitness landscape is constantly changing, as in Yaeger's *Polyworld*, and where *fitness* equals *reproduction*. These swimmers are situated autonomous agents, who mate autonomously.

This sets them apart from the organisms recently developed by Sims, which—though they are much more complex—are evolved using objective functions determined by design. This model, then, is more open-ended in terms of modeling reproduction. It also ties together the emergence of individual morphology and locomotion with population dynamics. Individual behaviors emerge which have meaning in the context of the dynamic ecosystems within which they evolve. For instance, particular kinds of swimming styles (i.e., aggressive and energy-wasteful, vs. slow and energy-efficient) can affect, and be affected, by environmental conditions (such as food supply—due to the rate at which swimmers must eat to replenish their energy).

In this world there are only swimmers, food bits, and the boundaries of the pond. A goal in this project has been to make the rules of the game as few and as straightforward as possible—motivated by the desire to minimize complexity in Design so as to focus on Emergence. Despite the model's brevity, the individual swimming techniques and overall group dynamics that emerge are sufficiently realistic, and a continual source of novelty. I define "swimming" in this paper to include any

activity that a swimmer does which enables it to move through the pond and to orient itself towards a goal which is itself moving. The swimmers have rudimentary perception and reactivity—which is hard-wired. They are not much on brains, but they have sufficiently versatile motor control systems and morphological schemes, for potentially many locomotion styles. The swimmers were designed with the ability to locate and choose food bits and mates that are within their view. They spend most of their lives "pursuing" these goals.

Key parameters were specified to supply an environment from which optimized behavior can emerge. These include:

- maximum life span of swimmers
- initial population size and area
- how far swimmers can see (view radius)
- water viscosity
- mass of swimmers
- rate at which food multiplies
- amount of food at the start of the simulation
- various levels of food energy exchange and loss
- various constraints in swimmer embryology

These settings are all explained in the following sections.

3.1 The Virtual Pond

The world in which the swimmers live is a two-dimensional virtual pond of water. Fluid flow is not modeled within the pond. Instead, interaction with water is reflected in the dynamics of a swimmer, as body parts stroke through the water and affect the swimmer's translational and angular velocities. Effects such as turbulence, wakes, vortices, etc., are not a part of this model. The pond is defined by a square boundary, which the swimmers cannot go beyond.

3.2 The Energy Cycle

During the running of the simulation, the distribution and dissipation of energy is computed throughout the pond. Energy originates from the food bits scattered around the pond. Food multiplies autonomously, at a slow rate—new bits appear at random positions in the pond at regular intervals. When a food bit is eaten by a swimmer, it is stored in the swimmer and increases its energy level by a predetermined amount. A swimmer uses up its stored energy in three ways:

1) Existing. A very small amount of energy is expended as a base-level metabolism.
2) Moving limbs. The more the swimmer exerts force against water, the more energy is used up (swimmers who exert more force against water become hungry more frequently than others).
3) Mating. The act of mating requires that a certain amount of stored energy be lost and transferred to offspring.

When a swimmer's energy level drops below a designated threshold, it assumes the *hungry* state, and quits whatever else it happens to be doing and begins to search for a nearby food bit. If it finds one, it will pursue this food bit. If it doesn't find a food bit, or cannot swim to a chosen food bit before its energy level reaches zero, it dies of hunger. If a swimmer is pursuing a food bit which another swimmer then eats, it looks for another food bit to eat.

4 Swimmers

Figure 1 illustrates the states a swimmer can assume, and the transitions from one state to another. Mating and eating are represented in the illustration with darker boxes. They are states that always occur in only one time step, and are immediately followed by the *looking-for-mate* state. The other states can last any length, depending on the situation. In addition to these states, any state can lead to *death* if 1) the swimmer's energy level drops to zero, and it starves, or 2) its age exceeds the swimmer maximum life span, and it dies of old age.

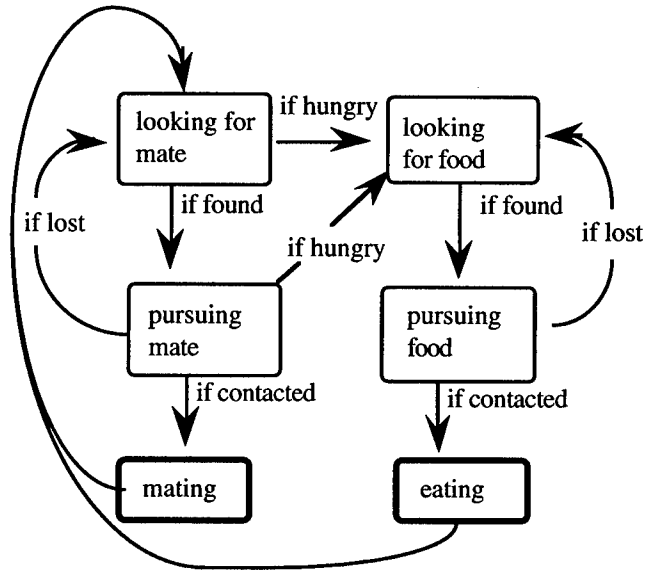


Figure 1. The states of a swimmer and the transitions between them

4.1 Perception

A swimmer's field of view covers 360 degrees, and so it can detect other swimmers or food from any direction. There is a limit to how far it can see—its view radius. A swimmer can recognize only the following:

- relative orientation of a food bit within its field of view
- relative orientation of a potential mate's head within its field of view
- the proportions of its favorite color in a potential mate

4.2 Mating

Gender difference is not modeled in the swimmer world. Any swimmer can mate with any other swimmer, with the exception of immediate family members (parents, siblings, and offspring)—this is one of the few constraints imposed upon the swimmers in the design of the model, to discourage pockets of stagnant inbreeding.

A chosen mate does not need to reciprocate in the mating pursuit for reproduction to happen. The chosen swimmer may have its own agenda, like searching for a mate of its own, or pursuing a food bit. This fact sometimes requires swimmers to pursue indifferent mates for long distances (which adds pressure for evolution of locomotion skill in the population).

There are no reproductive organs modeled in the swimmer world. Swimmers mate with their heads (this is all metaphysical anyway). The act of mating takes place when the heads of the swimmer and its chosen mate reach a distance less than one swimmer segment length (one "swimmer unit"). Mating results in at least one offspring, whose position is initialized directly between the parents, and whose orientation is random. Often, the pursuing swimmer will choose the same mate again in the next time step. In this case, another offspring is born from the same parents. This continues as long as the pursuing swimmer's energy is above the *hungry* threshold, and the chosen mate has enough energy to contribute its share of offspring energy. The number of consecutive offspring from the same parents typically range from one to four. There is no gestation period in this model.

Every swimmer is a phenotype, represented by a genotype consisting of 17 genes. Each gene is a real number ranging from 0.0 to 1.0, which is mapped to some real, integer, or Boolean value effecting the embryological process of constructing the phenotype. In a birth, the genotypes from both parents are copied and combined using random crossover points to create the offspring genotype. There is a chance of mutation in the offspring genotype, during reproduction. Mutation rate = 0.01.

4.3 Morphology

Swimmers are modeled as 2D figures comprised of interconnected line segments. There is one node (the head) from which a series of limbs emanate. The number of limbs can range from one to four. Each limb is divided into a series of segments, the number of which can range from two to four. All segments are of the same length.

The genotype was designed to utilize as few genes as possible, but generating a wide variety of phenotypes, in the spirit of Dawkins' *Blind Watchmaker* software (89). There are five morphology genes in the genotype. They affect the following phenotypic features:

- number of limbs
- number of segments per limb
- limb "straddle" (angle between consecutive limbs)
- joint angle (base angle per limb joint)
- limb reflection (when "on", this gene reverses all joint angles in alternate limbs)

Figure 2 shows twenty possible (un-evolved) morphologies. In this illustration, there is no autonomous motion affecting these forms so they appear unnaturally symmetrical.

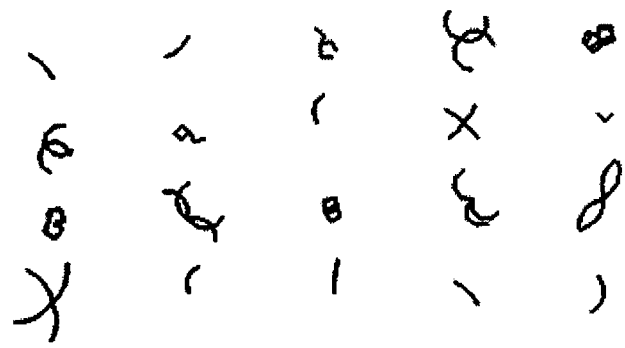


Figure 2. Morphologies of un-evolved swimmers

4.4 Motion Control

Each swimmer has an internal motor which drives a series of sinusoidal waves that bend its joints in myriad polyrhythms. In addition to this, the amplitudes and phases of these waves are modulated as a function of the orientation of a chosen food bit or mate, relative to the swimmer (for the purpose of turning). The nine genes dedicated to motion effect the following phenotypic features:

- rate (frequency of all wave motions)
- base amplitude for all motions
- limb to limb amplitude offset (adds or subtracts amplitude among consecutive limbs)
- segment to segment amplitude offset (added to the above offset)
- limb to limb phase shift (shifts phases of waves between limbs)
- segment to segment phase shift (added to the above phase shifts)
- turning amplitude (change in amplitude among consecutive limbs for turning)
- turning angle offset (explained below)
- turning phase (change in phase shifts among consecutive limbs for turning)

Each of these are real values which can be negative or positive.

4.5 Reactivity

In pursuit, a swimmer continually compares its axis of orientation to the relative direction of its goal, within its 360 degree visual field. The difference in these directions (an angular value, A) is used to change various parameters in the joint motions from limb to limb. This has the potential effect of causing the swimmer to affect its orientation (whether or not the swimmer rotates so as to

move *toward* its goal depends on many other phenotypic features). Three genes affect a swimmer's ability to turn (the last 3 in the list above). The amount that is added to the amplitudes of all joint motions in a limb for turning is equal to:

```
limb * cos((A + offset) * pi/180.0) * amp
+  (num_limbs-limb+1) * cos((A + offset)
* pi/180.0) * amp
```

where *limb* ranges from 1 to *number_of_limbs*, and *offset* is determined by the gene for *limb_turning_angle_offset*, and *amp* is determined by the gene for *limb_turning_amplitude*. The amount that the phases of waves in a limb's joint motions are shifted for turning is equal to:

```
limb * cos(A * pi/180.0) * turn_phase
```

where *turn_phase* is an angular value, controlled by the gene for *limb_turning_phase*.

5 Mate Preference

Three genes affect phenotypic features which come into play in times of mate choice:

- head color
- color shift from head to consecutive segments in all limbs
- favorite color in a potential mate

The first two genes control the coloration of the swimmer. Colors include the six primary and secondary colors of a painter's color wheel: red, orange, yellow, green, blue, and violet (violet being followed by red, in cyclical fashion). They are stored as consecutive integers in the model. An algorithm was designed which is not meant to model any particular biological scheme for animal coloration, but simply to generate phenotypic variety from a small number of genes. Coloration is determined by the following algorithm (expressed in C language).

```
for (L=1; L<=num_limbs; L++)
 {
  c = head_color;
  for (S=1; S<=num_segments; S++)
   {
    c = c + color_shift;
    segment(L)(S)color =
    red + ((int)(c*(float)(violet-red))
    % (violet-red));
   }
  }
```

where c is a positive real number, and head_color and color_shift are positive real numbers, determined by gene values. These two genes control the making a large variety of color patterns in swimmers, including solid colors.

The "favorite color" gene comes into affect when a swimmer is looking for an ideal mate. When sizing up the potential mates within its view, it chooses one who exhibits the largest amount of its favorite color, and, to a lesser extent, the two adjacent colors on the color wheel. For instance, a red-loving swimmer will tend to choose mates having lots of red, violet, and orange. Red would have twice the "strength" as violet or orange, in enticing this swimmer's lust. Swimmers exhibiting none of these colors would not be chosen by the red-loving swimmer.

Amount, not *percentage* of favorite color in a mate is perceived. Thus, more limbs can potentially have a larger effect—the expectation is that sexual selection can have an effect on the evolution of body size.

6 The Physics of Motion

Forward dynamics is used to approximate the key physical affects required for a swimmer to do its thing, most importantly - the interplay between momentum generated by sweeping body parts, and drag, and the effects of torque in generating angular acceleration.

Swimmer's bodies are approximated as 2D rigid bodies composed of line segments, which can change internally via the autonomous bending of joints. A swimmer's internal geometry cannot be changed by outside forces. In a 3D version of this world, these line segments would roughly correspond to thin cylinders of equal size, shape, and mass. A swimmer's mass equals the sum of all limb segment lengths times a *mass* constant.

As stated before, fluid flow is not calculated in this model—the fluid is assumed to be fixed in a world frame. Its job is simply to supply friction to give the swimmers something to work against. Proportional to a friction constant, the motion of each limb segment creates a force proportional to the sine of the angle between the segment orientation and the segment velocity.

Angular momentum is affected by torques created by each limb segment's force, in relation to the center of mass. In each time step, the accumulation of all linear and angular forces generated by individual segment motions are summed up to create net linear and angular accelerations.

This physics model is not comprehensive, but sufficient for the purposes of this study. And it allows for both paddling-type motions and undulating-type motions to be affective in propelling the body, assuming there is minimal drag caused by other body parts. It also supplies a variable corresponding to the magnitude of total forces exerted on the water, which is used to approximate the expenditure of energy in a swimmer.

7 Running Simulations

At the start of a typical simulation, a population of 1000 swimmers are initialized with random genotypes. Their phenotypes are positioned and oriented randomly within a disk-shaped region, called the *Garden of Eden*, located in the middle of the pond. They are packed densely to give them an initial advantage in getting to each other. Each swimmer's age is initialized to zero. Energy levels are set equal to what two parents would contribute to an offspring during reproduction. The Garden of Eden is also initialized densely with 3000 (the maximum) food bits, to give early swimmers an added advantage.

7.1 Observations

Swimmers are born hungry (energy at birth is fairly low), and so immediately begin to look for food bits. At initialization, many swimmers are born lucky and happen to be initialized close enough to a food bit that they can eat it immediately. These swimmers then begin looking for mates. Many first generation swimmers are destined to die of hunger, since they do not have enough locomotion skill to reach a chosen food bit. Among those who are able to eat, a small percentage are also able to mate. The result is a high death rate in the first 100,000 time steps or so. Areas in the pond with swimmers whose randomly-initialized genotypes give them a slight advantage will reproduce, thus creating areas of higher birth rate. These areas often blossom into communities, and begin foraging through the fertile Garden of Eden.

Figures 3a through 3d show the distribution of swimmers and food at time steps 1, 130,000, 175,000, and 225,000, respectively. 3a and 3b are slightly magnified views of the pond center: 3c and 3d show the entire pond. Food bits can be seen accumulating in the pond, which later become sparse as the swimmer population grows, consuming more food.

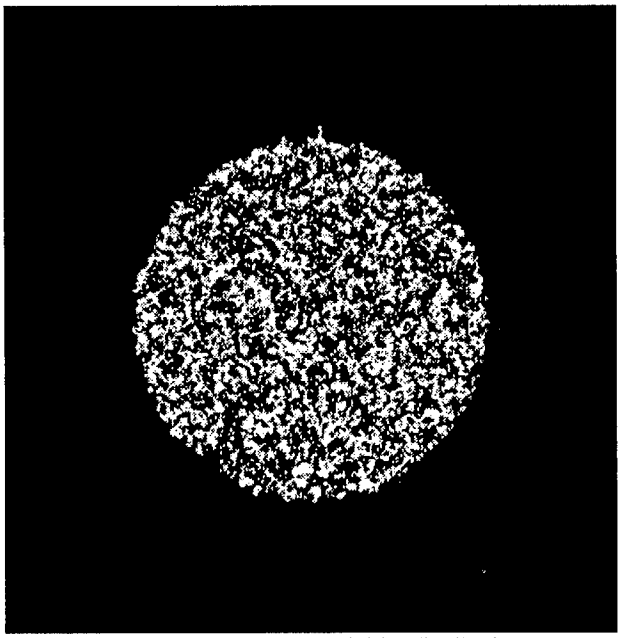


Figure 3a. The Garden of Eden - initial distribution

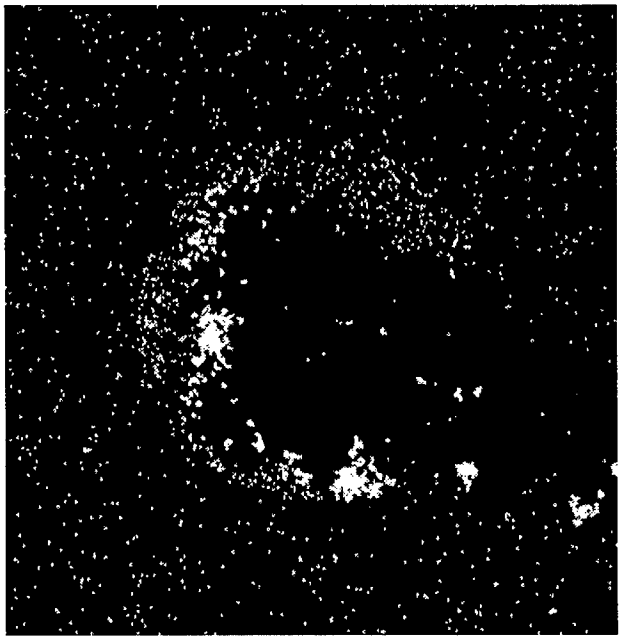


Figure 3b. At 130,000 time steps

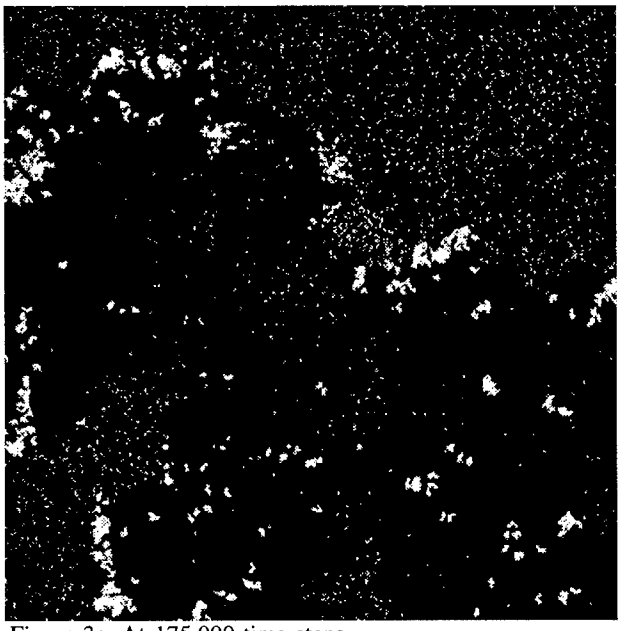


Figure 3c. At 175,000 time steps

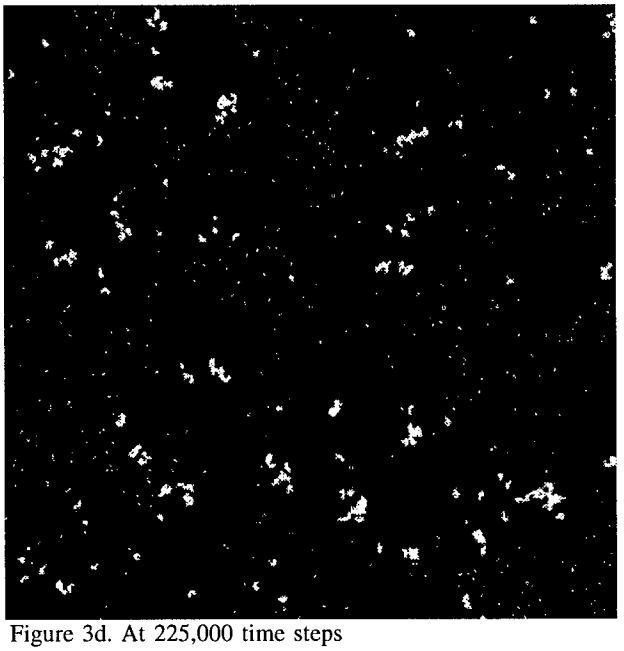


Figure 3d. At 225,000 time steps

The following sequence of events occurs for most of the simulation runs:

1) A period of occasional random births and deaths, mostly deaths.
2) Among the high death rate, a few pockets of high birth rate appear, indicating presence of successful swimmers.
3) Sudden drop in population count due to first generation swimmers dying of old age (in some simulation runs, the population dies off entirely soon after this point).
4) Increase in population, indicating that some remaining swimmers have inherited locomotion skills and are now reproducing.
5) Groups of genetically related swimmers appearing in isolated areas.
6) Distinct waves of foraging as groups of swimmers move towards areas of high food count, leaving no food in their wake.
7) Population increase in the swimmers, and drop in food count as swimmers mate and eat at higher rate.
8) Consequent drop in swimmer population due to low food supply, with left over swimmers able to eat and mate at a higher rate.
9) Takeover of pond by one or two color groups.
10) After some oscillations, food count vs. swimmer count reach general stability, with occasional new color groups forming and old ones dying off.
11) Increased speed, turning ability, and energy efficiency in swimming usually results over very long periods (in some simulations, the population eventually evolves into graceful one-limbed or two-limbed undulating forms).

Figure 4 shows a typical swimmer vs. food population count for a period of 300,000 time steps. Numbers representing key stages in the above list are annotated at bottom.

Maximum swimmer count in this run was 1000. Maximum food count was 3000, indicated by the ceiling at top. The food count is at its maximum value at the start of the simulation, but due to lucky swimmers eating bits in the first time step, the food count appears to begin lower.

7.2 Turning

As hoped, more optimized locomotion and anatomy emerge in all simulations. Most paths traced by swimmers tend to be spirals of varying types, some paths spiraling out, some in. Some paths create spirals within spirals. The turning mechanism of evolved swimmers tends to correct counterproductive spiraling paths. These stimulus-response behaviors are somewhat reminiscent of the paths exhibited in the classic "Vehicles" of Braitenberg [84].

At the start of a simulation, approximately half of the swimmers (those who can travel at all) spiral away from their chosen mates. Among the others, a few are able to close in on their chosen mates, before getting hungry and sidetracked by a nearby food bit. Indirect paths sometimes

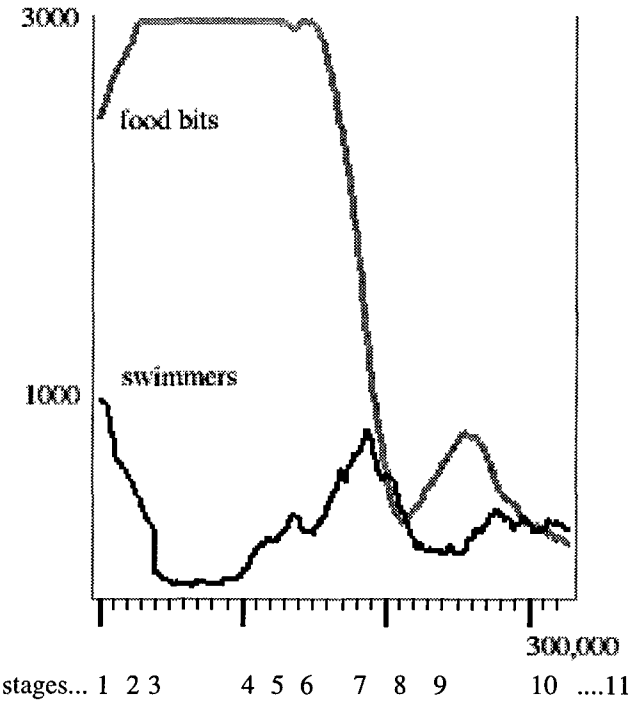


Figure 4. Swimmer vs. food populations for 300,000 steps

resemble mating rituals, and can be amusing to watch—but it only goes this far: these behaviors tend to die out, since more direct paths result in quicker reproduction.

The evolution of turning skill involves some quirky behaviors. Some populations acquire the ability to turn in only one direction, and so must do more work in order to orient towards a goal. Some populations have swimmers who actually swim in two modes: backwards and forwards—the two modes being qualitatively different than each other, and one usually being more efficient than the other.

7.3 Energy Efficiency

Since energy expended in swimming is proportional to the total magnitude of force a swimmer directs against water, swimmers whose motions are erratic, or whose strokes cancel each other out get hungry often, and so have to spend most of their lives pursing food. They also risk a higher chance of dying of hunger. Thus, they do not mate often, and are likely to die out completely—especially in the presence of more energy-efficient communities in the pond, with their own demands on food. "More efficient swimming" means: 1) closing in on a goal at a higher velocity and 2) expending less energy—than the average swimmer. These terms are not defined numerically in this model, but qualitatively—numerical measurements of swimming efficiency are likely to be made in a subsequent study. The following figures illustrate some of these qualitative observations.

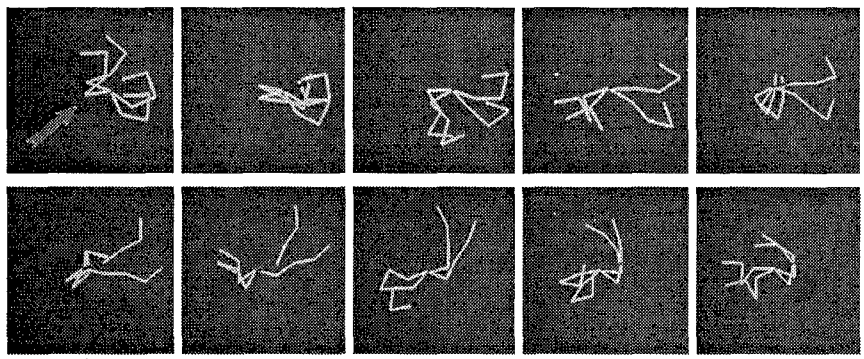


Figure 5. A representative swimmer from a population of 4-limbed, inefficient swimmers.

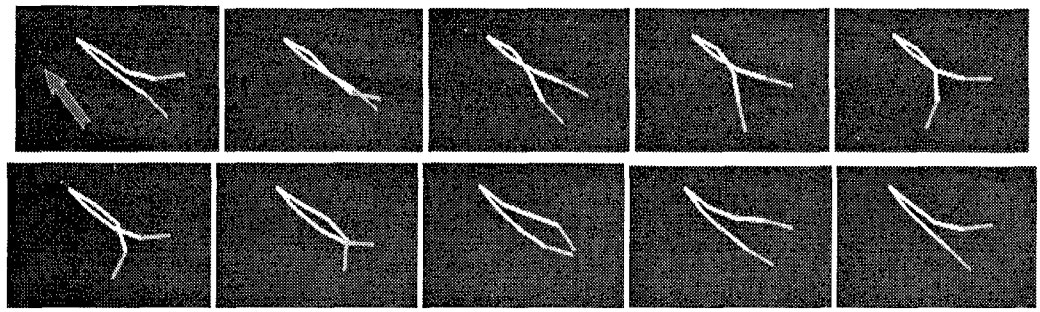


Figure 6. A representative swimmer from a population of 2-limbed, streamlined, undulating swimmers.

Figure 5 shows a representative swimmer in a color group in the pond comprised of *inefficient* swimmers, during the earlier stages of a simulation run. Their morphologies include the maximum body-segment count. In this illustration, ten frames from a portion of the characteristic stroke of the swimmer is shown. The swimmer's goal is located at upper right, as indicated by the arrow in the first frame. The two outstretched limbs in frames 6-9 are responsible for most of the locomotive work. The motions of the other two limbs are erratic, probably encumbering.

Figure 6 shows a swimmer with a more streamlined swimming style, exhibited by a color group in the pond contemporary with the group described above (these two communities did not interbreed due to emergent differences in mate preference). In this example, swimming style is graceful and undulating. In the simulation in which the two swimming styles illustrated above were present, the less efficient swimmers were able to sustain their population for more than 100,000 time steps, despite the increasing size of the more efficient group—which eventually took over the pond.

In some rare simulation runs, very aggressive and energy-wasteful swimmers manage to take over the pond. Swimmers of these populations are, consequently, hungry for a larger percentage of their lives, and eat at a high rate, yet are still are able to mate at a sufficient rate to keep the population at a sustainable size. In these populations, food in the pond is sparse, and it is common to see dozens of swimmers racing to try to get to one precious food bit, as shown in Figure 7. In all cases, one swimmer eats the bit, then all the others immediately dissipate to pursue other food bits. In this time-lapse photograph, the direction of the swimmer's goal is shown as a white dash located on its head, oriented at the goal. Multiple animation frames are superimposed in this image to indicate limb movement.

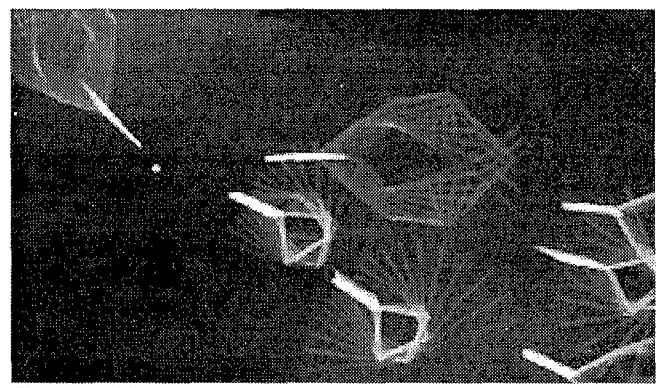


Figure 7. Time-lapse photo of seven aggressive swimmers racing to get to a food bit

Figure 8 shows the characteristic stroke of a swimmer from an advanced population (715,000 time steps). This population has learned to maximize its stroke effect (frame 7), and to avoid drag in follow-through (frames 9, 1).

Two distinct techniques for swimming are noted to emerge: paddling and undulating (and some combinations of both). Figure 5 illustrates paddling behavior, and Figure 6 illustrates a kind of undulating motion. It may have been hubris which originally caused me to anticipate more anthropomorphic limb-stroking techniques to emerge. I was initially disappointed to see many advanced populations reduced to single-limbed undulating eel-like figures (and all that potential anatomy I programmed in was unused!). But in fact, the sea itself illustrates that the most optimized form of underwater travel involves streamlined bodies and sinuous motion.

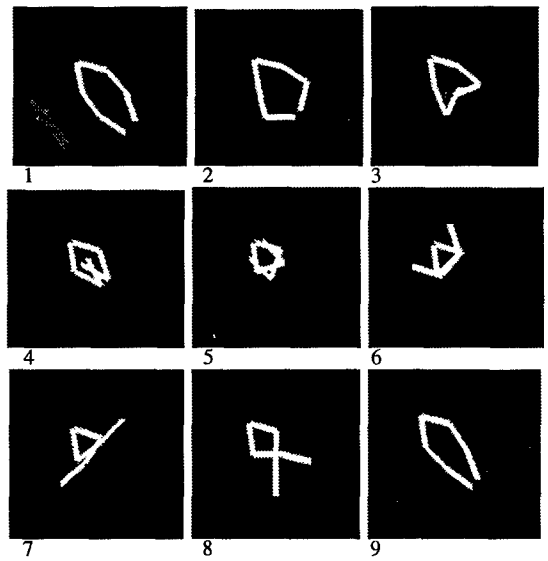


Figure 8. Nine frames of the stroke cycle of a swimmer from an advanced population

7.4 Results of Sexual Selection

The effects of mate preference are not noticeable in early stages of evolution. Since both body coloration and color preference in mates are initialized randomly, there is no meaningful correlation in these factors at first among swimmers in the population. A swimmer who loves red swimmers may as easily consist of greens and yellows as violets and reds. But small, chance pockets of correlation in these two factors can stimulate higher growth rate in areas of the pond. Eventually, color preferences begin to correlate with actual body colors.

It is hypothesized that this happens for two reasons: first, offspring can inherit the colors from the chosen mate along with the preference for that color in the chooser. And at no time can a mating occur when there is NO correlation between a swimmer's color preference and its own body color. Secondly, since this is a spatial model, proto-communities can emerge in local areas of the pond consisting of swimmers who, by virtue of similar preferences—*and* their proximity—tend to mate within their group only. This inter-group mating then reinforces the *group identity* further.

It's not possible to show color in these illustrations, but some of the white clumps seen in Figures 3c and 3d are distinctly different in color. In a spatial evolutionary model such as this, the "clumpiness" of phenotype space is often the result of geographic separation, but in this model, differing mate preferences among color groups further encourage phenotypic clumps. This is indicated by the presence of different color groups converging on a common food-rich area but not interbreeding.

To analyze the effects of sexual selection in generating isolated groups, a correlation factor *C* is defined: a swimmer whose body coloration is homogeneous and whose mate color preference is identical with its body color has a correlation factor of 1. A swimmer whose body coloration has no colors identical to or adjacent to its mate color preference has a correlation factor of 0. All possible phenotypes have correlation factors lying between 0 and 1.

The coherence (propensity for swimmers to go after their own kind) of a group of swimmers is defined by its average correlation factor. That group is also defined as being homogenous in mate preference. Figure 9 shows a plot of correlation factors over the span of 240,000 time steps in a simulation run, with a typical outcome. Every 1000 frames are plotted. It shows correlation factors for each of six groups of swimmers, categorized by the six possible mate color preferences—labeled R, O, Y, G, B and V. Groups comprised of less than 10 swimmers are not plotted. In this plot, R is not shown.

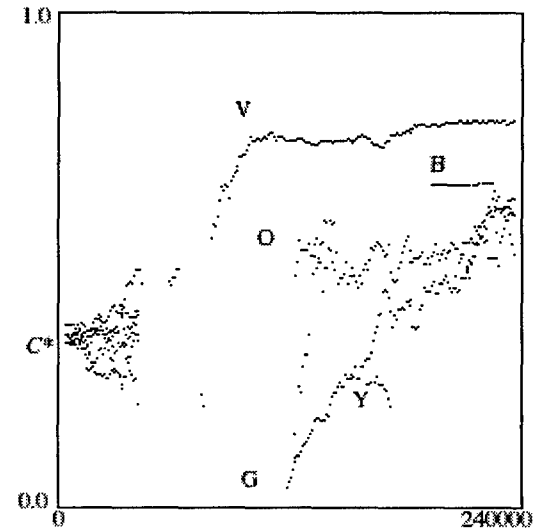


Figure 9. average correlation of mate color preference with swimmer's own body color: five mate preference groups are plotted over time.

At initialization, correlations are around $C^* = 0.333$ (the theoretical average correlation when genotypes are random—given the model's parameters for color attraction). In this simulation, the whole population size increases around time step 100,000, as the violet group V emerges. This group remains the dominant one throughout the simulation. At one point, a mutation in one area of the pond creates a group of green-loving violet swimmers G. Its average correlation factor is below C^*. However, this group persists for many thousands of time steps, and eventually increases the amount of blue in its swimmers' bodies (a color which green-loving swimmers respond to), thus increasing its average correlation factor steadily over time, as shown. Three other groups—O, Y, and B, also emerge, with O persisting.

7.5 Peacock Tail Effect

It appears that sexual selection counteracts a natural selection tendency for streamlined, energy-efficient swimmers with few limbs, due to the fact that swimmers with more limbs can potentially flaunt more colors to admirers. Experiments with simulations run with and without sexual selection support this observation.

8 Future Work

The inclusion of an organism's articulated motion dynamics in a spatial evolutionary model suggests the possibility of more emergent complexity and realism in artificial life simulations. This could be achieved to a greater extent than shown here by combining the impressive results of creatures evolved "a la Sims", within "Yaeger worlds", to generate virtual ecosystems in which complex emergent behavior can be witnessed on many scales, from the global interactions of species populations down to the wiggling of a tail. A more open-ended genotype representation, such as accomplished with genetic programming, would allow much more novelty to emerge within the swimmer world.

Also, sexual selection could be extended to include preferences for qualities of motion and morphology. Observing the tradeoffs and compromises between "sexiness" and efficient swimming could make for an interesting study.

A strong motivation in this project is to help bring the knowledge and excitement of this type of work into the realm of non-specialists, through instructional and *inspirational* interactive media. Future development in this regard is in the works.

9 Conclusion

This project demonstrates how a simulation can generate emergent behavior on multiple scales in a simple virtual ecosystem, in which the small details of emergent locomotive behavior in individual organisms can affect, and be affected by, large-scale global dynamics. By combining recent evolutionary computation techniques for creating articulated virtual creatures with spatial, open-ended reproduction models, this work suggests that articulated movement in an artificial organism can come to have "meaning" in the context of the world in which it was born.

This work also demonstrates a rudimentary example of sexual selection, and how this can influence the distribution of phenotypes in a spatial model.

Acknowledgments

Thanks to Rocket Scientist Bryan Galdrikian for his help in many aspects of this work. Thanks to Bill Davis and the rest of the crew at Rocket Science Games, Inc. for use of facilities, and for supporting my need to continue making wiggly things. Thanks to Mark Weissman for help in videotape production. And special thanks to Nuala Creed.

References

Badler, N., Barsky, B., Zeltzer, D. *Making Them Move.* Morgan Kaufmann, 1991.

Braitenberg, Valentino. *Vehicles: Experiments in Synthetic Psychology.* MIT Press, Cambridge. Mass. 1984.

Dawkins, R. The Evolution of Evolvability. *Artificial Life* Addison-Wesley 1989 pages 201-220

Goldberg, D. *Genetic Algorithms in Search, Optimization, & Machine Learning.* Addison-Wesley, 1989

Holland, J. *Adaptation in Natural and Artificial Systems.* University of Michigan Press, Ann Arbor. 1975

Koza, J., *Genetic Programming: on the Programming of Computers by Means of Natural Selection.* MIT Press, 92

Langton, Christopher, editor. *Artificial Life* Addison-Wesley 1989

Ngo, Thomas J. and Marks, Joe. Spacetime Constraints Revisited. *Computer Graphics* . pp. 343-350. 1993

Ray, T. An Approach to the Synthesis of Life. *Artificial Life II,* Ed Langton, 1991, 371-408

Reynolds, Craig. Flocks, Herds, and Schools: A Distributed Behavioral Model. *Computer Graphics,* Vol 21, Number 4, July, 1987

Sims, K. Evolving 3D Morphology and Behavior by Competition. *Artificial Life IV.* MIT Press 1994

Sims, K. Evolving Virtual Creatures. *Computer Graphics.* SIGGRAPH Proceedings pp. 15-22 1994

Terzopoulis, D., Tu, X., and Grzeszczuk, R. Artificial Fishes with Autonomous Locomotion, Perception, Behavior, and Learning in a Simulated Physical World. *Artificial Life IV.* MIT Press, 1994

Todd, P.M., and Miller, G.F. On the sympatric origin of species: Mercurial mating in the Quick-silver model. In R.K. Belew and L.B. Booker (Eds.), Proceedings of the Fourth International Conference on Genetic Algorithms (pages 547-554). Morgan Kaufmann. 1991

Ventrella, J. "Walker" video demonstrated at Artificial Life III conference, and an unpublished paper, produced at Advanced Graphics Research Lab of Syracuse University, NY 1990 (contact: ventrell@media.mit.edu)

Ventrella, J. Explorations in the Emergence of Morphology and Locomotion Behavior in Animated Figures. *Artificial Life IV.* MIT Press 1994

Witkin, A. and Kass, M. Spacetime Constraints. *Computer Graphics.* 22(4): 159-168 Aug. 1988

Yaeger, L. Computational Genetics, Physiology, Metabolism, Neural Systems, Learning, Vision, and Behavior or PolyWorld: Life in a New Context. *Artificial Life III.* Ed Langton, 1994

Zeltzer, David. Task Level Graphical Simulation: Abstraction, Representation, and Control. from *Making Them Move.* ed Badler. 1991

COEVOLUTION

Cooperative Versus Competitive System Elements in Coevolutionary Systems

Robert E. Smith and H. B. Cribbs III
Department of Engineering Science and Mechanics
The University of Alabama
Box 870278
Tuscaloosa, Alabama 35487
Phone: (205) 348-1618
Fax: (205) 348-7240
Email: rob@comec4.mh.ua.edu
brown@galab3.mh.ua.edu

Abstract

This paper examines systems that have coevolved elements. For any given environmental state, the overall action of such systems can be formulated in one of two ways: the system's elements can cooperate to jointly formulate an action, or the elements can compete, with a winner formulating an action. In both cases, underlying coevolution may be necessary to create a group of elements that can implement a complete action repertoire. However, one would expect that the emergent relationships between elements would differ between competitive and cooperative systems. These differences have important implications for both natural and artificial adaptive systems. This paper examines this issue by using neural networks that evolve under the action of a genetic algorithm. These neural networks are functionally similar to learning classifier systems. Cooperative and competitive versions of these networks are evaluated in a simulated mobile robot test problem. Experimental results lead us to the conjecture that certain symbiotic relationships between elements (e.g., default hierarchies and chains) develop more easily in cooperative systems than in competitive systems. Implications of this conjecture, and future directions of inquiry are discussed.

Keywords: coevolutionary models, evolutionary computation, emergent structures and behaviors, autonomous robots, action selection

1. Introduction

Coevolutionary adaptive systems consist of a diverse, evolving group of elements that act together to perform useful computations, or execute effective behaviors. These systems are common in nature (including ecologies and organic brains), in social systems (including economic and political systems) and in artificial adaptive systems (including rule-based, neural, and a variety of other types of AI and Alife systems).

There are two distinct forms of coevolutionary systems:

- Those where each action is formulated by a consensus of elements through *cooperation*, and
- Those where each action is formulated by a single element, selected by *competition* to cope with the current situation.

The distinction is made more clear by considering the following social system analogy. Assume a decision must be made by a group of individuals. One group decision strategy is to vote on each decision option, and select according to vote tallies. This corresponds to cooperative action formulation. Another strategy is to select a representative (based on any number of competitive criteria) to serve as a champion or elected official who will make the decision for the group. This corresponds to competitive action formulation.

In the realm of artificial adaptive systems, both types of behaviors are also represented. Typical neural networks are characterized by the cooperative formation of actions, typically through the summation of hidden layer neuron activations at an output layer. Typical rule-based adaptive systems (like *learning classifier systems* (LCSs)) are characterized by competitive formation of actions, through a conflict resolution mechanism that picks a single winning rule for any given situation.

In natural systems, the distinctions between the two types of action formulation are somewhat less pronounced. However, species that competitively select champions (or leaders) are certainly observed. Cooperative selection of group actions can be observed in any group where individual action has an accumulated effect on the entire species or subpopulation.

It is important to note that both cooperative and competitive systems may require coevolution. Even in a

competitive system, several elements may need to exist and coevolve to one another to form a complete action repertoire. Such elements serve as champions for different environmental situations. However, one would expect that the emergent computational structures that evolve in cooperative systems would differ from those that evolve in competitive systems. This paper examines some of these differences.

The particular systems employed in this paper's examinations are based on an analogy between LCSs and neural networks presented by Smith and Cribbs (1994). The systems are neural networks whose structures evolve under the action of a *genetic algorithm* (GA). However, unlike the system presented in the (1994) paper, the systems used here have been adapted to perform Q-learning in a reinforcement learning environment. The particular test environment used is a simulated autonomous robot environment.

Although the experiments presented here are artificial intelligence systems, it is hoped that the implications may extend to a variety of other adaptive systems, including Alife and organic adaptation. In particular, results indicate certain advantages to cooperative systems. The systems, experiments, and implications of these results are discussed further in the remainder of this paper.

2. Problem Setting and Experimental Techniques

To introduce the systems and experiments used in this paper, it is important to provide a formal setting for the problems and techniques employed. The setting used will be *reinforcement learning*. However, it is important to note that the general conclusions may have implications outside this class of problems and techniques.

2.1 Reinforcement Learning

In reinforcement learning, a control system must submit signals to an initially unknown plant. The plant provides a signal related to its internal state, but this signal is often a noisy or incomplete indication of the environment's actual state. The plant also provides an error signal, which it can use as a basis for optimizing its control actions. However, this signal does not directly indicate error in the actions the controller takes. Rather, it offers some measure of error in the state of the environment. This error signal is often called punishment, cost, or reward. Regardless of its form, this signal may also be noisy, or time delayed. Note that this reinforcement learning problem accurately characterizes the situation faced by many autonomous, adaptive systems in unknown environments.

To represent a strategy for picking actions, consider the following approach. For each action u associated with a state i, assign a value $Q(i,u)$. One can translate this set of values into a definite strategy in the following fashion. At each time step, select the action associated with the best Q. Under such a strategy, Watkin's *Q-learning* algorithm (1989) yields optimal Q-values in restricted situations.

A key step in any learning system is the representation or storage of a strategy. The previous discussion suggests Q-values as a method for storing strategies. However, a primary difficulty in the Q value approach is that it requires storage of a separate value for each state/action pair. The system used in this paper can be seen as a technique for simplifying the storage of Q values. It is based on LCSs and neural networks, which are discussed in the following sections.

2.2 Learning Classifier Systems

Consider the following method for representing a state/action pair in a reinforcement learning problem: encode a state in binary, and couple it to an action, which is also encoded in binary. In other words, the string

0 1 1 0 / 0 1 0

represents one of sixteen states and one of 8 actions. This string can also be seen as a rule that says "IF in state 0 1 1 0, THEN take action 0 1 0." In a *learning classifier system* (LCS), such a rule is called a classifier (Holland & Reitman, 1978). One can easily associate a Q value, or other performance measures, with any given classifier.

Now consider generalizing over actions by introducing a "don't care" character (#) into the state portion of a classifier. In other words, the string

1 1 # / 0 1 0

is a rule that says "IF in state 0 1 1 0 OR state 0 1 1 1 OR state 1 1 1 0 OR state 1 1 1 1, THEN take action 0 1 0." The introduction of this generality allows an LCS to represent groups of states and associated actions as classifiers.

Clearly, one cannot (in general) store and evaluate all possible generalized classifiers in a realistically sized problem. The selection of an appropriate set of generalized classifiers is equivalent to the search for appropriate groups of states that can be treated with the same Q values. When viewed as classifiers, this is clearly a combinatorial search problem. One must find combinations of 1s, 0s, and #s for which reliable, useful Q values can be calculated by the Q-learning procedure. GAs are the primary mechanism used in LCSs to search for useful rules.

One should note that the search problem involved in finding groups of states for Q values is not a typical optimization problem. In this problem, one must find a *set* of classifiers. This set is of unknown size. By treating each classifier as an individual in the search process, and

searching for a co-adapted ecology of such classifiers, the GA is naturally suited to this search problem.

Note that Q-learning is not the most common method of credit assignment in LCSs. The most common method is the *bucket brigade algorithm* for updating a classifier performance statistic called *strength*. However, the bucket brigade is somewhat similar to Q-learning (Holland, Holyoak, Nisbett & Thagard, 1986; Goldberg, 1989; Wilson, 1985).

2.3 Neural Networks and Q-learning

Another practical approach to storing Q values is to store a function that maps a state to a set of Q values (one for each possible action). Given the constraints of reinforcement learning, this function must be obtained through on-line learning. Neural networks are one approach to this function learning problem. The input to such a network is the current state of the environment. This state can be represented in binary, as in the LCS, or in some other form. There is one output for each possible action. When a state is submitted to the network as an input, the outputs are the Q-values associated with each action. To teach the neural network the appropriate mapping from states to Q values, one can use the backpropagation algorithm (Smith & Cribbs, 1996).

2.4 LCSs and Neural Networks

Many comparisons have been made between LCSs and neural networks. A concise analogy between the two is offered in Smith and Cribbs (1994). This analogy is related to work presented by Wilson (1990). This paper extends that analogy to a GA-shaped neural network that learns Q-values. Consider Figure 1.

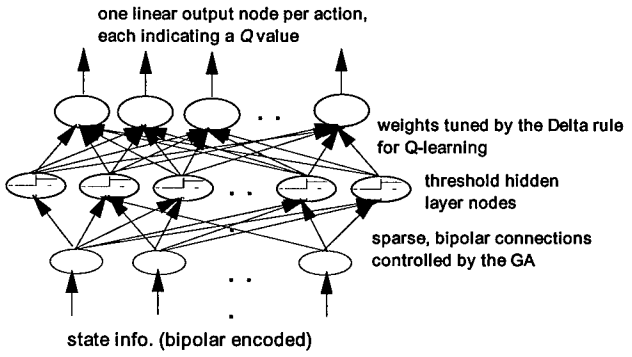


Figure 1. A neural, genetics-based cooperative classifier system that incorporates Q-learning. Note that *bipolar* weights and inputs are defined to be either -1 or +1.

Note that threshold hidden neurons are used here, as in Smith and Cribbs (1994). Linear output neurons are used to output a set of Q values. A gradient descent update is used to change weights in the hidden-to-output layer. The error measure used in this update is based on Q-learning. The bipolar (i.e., {-1,1}) weights and connections in the input-to-hidden layer are shaped by the GA. Each node in the hidden layer is unique in its input connection pattern, and represents a group of identical individuals in the GA population. Therefore, the GA can effectively vary the number of hidden layer nodes from 1 to the GA population size by varying the number of unique population representatives. This is similar to the *macroclassifier* concept used by Wilson (1995).

Note that this system's incorporation of Q-learning into a LCS, and its neural analogy draw on several other ideas also suggested by Wilson (1994; 1995). A significant addition to past LCS ideas is the hidden-to-output layer connectivity. Each hidden layer neuron is connected to a Q-value output neuron for every action. Therefore, each hidden layer neuron is essentially equivalent to a classifier with a complete set of actions (or a set of classifiers, one for each action, with similar conditions). Each action has an associated Q-value. How these Q-values are computed depends on the nature of the hidden layer. The layer can be competitive, as in the LCS analogy, or cooperative, as in traditional neural networks. In the cooperative system, the outputs of all hidden layer neurons are summed at the output layer. In addition, all neurons with non-zero output (i.e., those which match the current input) are updated. In the competitive system, a single, winning hidden layer neuron is selected, and its outputs alone are sent to the output layer. Only the winning neuron is updated.

The differences in emergent, coevolutionary effects in competitive and cooperative versions of this system are examined in an autonomous robot test environment in the following sections.

3. A Simulated Robot Test Environment

To test the cooperative and competitive systems described above, we have used a simulated, mobile robot test environment similar to that used by Mahadevan and Connell (1992). Although the simulation used here makes no claim of being realistic with regard to a hardware robot, they do adequately serve the purpose of comparing cooperative and competitive coevolutionary systems.

The simulated robot has a radius of 6 inches. The robot's 4 sensors are arranged at plus and minus 15 and 30 degrees on either side of the forward movement direction. The sensors are rather narrow and myopic. Each sensor indicates if an object is closer than 36 inches from the robot's center, along the sensor's line-of-sight. If the object is between 18 inches and 36 inches from the robot's center, the sensor indicates it is "far" from the robot. If the object is closer than 18 inches from the robot's center, the sensor indicates it is "near" to the robot. The robot also has two additional sensors that indicate whether it is bumping into an object,

and whether it has become stuck. The robot is stuck if it bumps for two or more consecutive time steps.

At each time step, the robot can take one of 5 actions: it can move forward 6 inches, or it can turn 22 degrees or 45 degrees to the left or right.

The simulated robot is placed in a simulated, rectangular room with polygonal obstacles. In the experiments presented here, the robot's room is 288 inches square. The robot is represented by a small circle. Its direction is indicated by a small dot on its front edge.

In the experiments presented here, the robot's goal is to simply explore as much of its environment as possible, without bumping into any obstacles. Experiments are presented with two different arrangements of polygonal obstacles. The first environment is a relatively unconstrained environment, with lots of open space. In this environment, if the robot bumps, it is restarted at one of 10 random, valid locations in the room, with a random orientation. The second environment has a large rectangular obstacle that constrains the robot to a narrow hallway. If the robot bumps in this environment, it is restarted at a fixed location in the hallway.

To provide positive feedback for exploration, the room is divided into a grid of 1-square-foot cells. When the robot's center enters a cell, it receives the reward available in that cell. Initially, each cell contains 500 points of reward. After a cell is entered, that cell's reward supply is depleted. After the robot leaves a cell, the reward regenerates 5 points per time step, reaching its maximum of 500 in 100 time steps.

The robot is punished 10000 (-10000 reward) for bumping. Turning in place costs 100 points of reward (-100 reward). Moving straight ahead in a cell where reward has been consumed incurs no penalty or reward.

4. Cooperative System Experiments

In the following sections, the LCS-based neural network system discussed above will be employed with a cooperative hidden layer in the simulated robot test environment. Experimental details are included below.

4.1 Experimental Details

This study employs a *block* GA, similar to that used in Smith and Cribbs (1994). However, in this study, the GA is *triggered* by the robot's actions, as opposed to activating at pre-set time intervals. The GA acts on a block of 10 classifier neurons (that are selected based on *fitness*) after the robot bumps 15 times. This allows the GA to respond when needed to refine the neural network. The 10 selected classifier neurons are recombined and mutated to form 10 new classifier neurons. These neurons replace 10 neurons that are selected for their lack of fitness.

The GA in this study uses two-point crossover with probability p_c, and a variation of point mutation. The mutation operator acts on a connection with probability p_m. When active, this mutation operator changes -1s to +1s, and vice versa, with probability $1-p_{\#}$, and changes either to a 0 with probability $p_{\#}$.

Clearly, the fitness of a neuron is critical to the GA's operation. The fitness function used here is quite similar to that presented in Smith and Cribbs (1994). However, in that *supervised* learning study, the fitness function was based on the following

[FIRE AND NOT(ERROR)]+[FIRE OR NOT(ERROR)]

where FIRE and ERROR are binary variables indicating whether the given neuron fired, and whether the single output neuron erred in its output. This calculation, summed over all input cases, indicates the correlation of the given neuron's firing pattern to error.

While the firing state of each neuron is readily available in the current reinforcement learning study, error is not. As was described earlier, reinforcement learning problems provide feedback as reward or punishment, as opposed to error in the control system's output. For this study, we substitute reward for error in a digital sense, i.e., a positive reward exists or it does not. Two basic terms are included in the fitness measure:

base = FIRE OR REWARD
extra = FIRE AND REWARD

where REWARD is the digital interpretation of reward discussed above. Fitness is then calculated as

fitness = (base+extra)*decisiveness

where decisiveness is a factor introduced to encourage hidden layer neurons that decisively influence the distribution of Q values. In this study it is possible to produce rules that fire with essentially the same weight on all actions. Such neurons advocate all actions, and are unnecessary. Decisiveness emphasizes rules that advocate a select set of actions instead of advocating all actions. This study used the variance of all the hidden-to-output weights for the given neuron (scaled between zero and one) as a measure of decisiveness.

In the experiments presented here, we select an action at each time step by adding normal noise to all the Q values, and selecting the action with the largest value. This *noisy auction* is a simple technique that has been used in other

LCS studies (Goldberg, 1989). The neural network used in this study uses gradient descent with an adaptive learning rate update scheme to update hidden-to-output layer weights. Details of this scheme are given in (Vogl, Mangis, Rigler, Zink & Alkon, 1988). As a basis for learning rate adaptation, windowed averages of Q-value updates are used.

Parameters for the experiments presented here are given in Table 1.

Parameter	Value
Q-learning	
α (Update Rate)	0.2
γ (Discount Rate)	0.75
Noise Amplitude	20
Neural Network	
Initial Learning Rate	0.016
Learning Rate Increase Rate	1.05
Learning Rate Decrease Rate	0.7
Learning Rate Adaptation Threshold	1.04
Averaging Window	250
Genetic Algorithm	
Block Size	10
p_c	0.8
p_m	0.01
$p_\#$	0.1
Population Size	100
Trigger	15

Table 1: Experimental parameters for the cooperative system.

4.2 *Cooperative Results*

In typical experiments with the suggested cooperative system and the relatively unconstrained environment (Figure 2), the robot took approximately 1500 time steps to evolve an effective behavior. However, GA refinement of the neural network continued throughout the experiment. At the end of this evolution, the neural network had 16 coevolved hidden layer neurons, and approximately 24 percent connectivity between the input and hidden layers. The robot's behavior at this stage is very effective. Depending on the robot's starting point and configuration, it can take approximately 500 steps before bumping into an obstacle. The eventual collision is typically due to the robot grazing the corner of a polygonal obstacle, and is probably due to the extremely limited sensors. Since the simulated sensors only react to objects in the line of sight, grazing objects cannot be avoided.

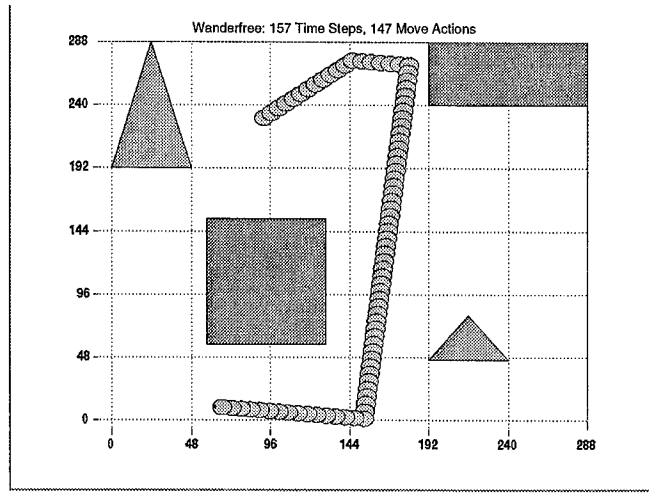


Figure 2. Experimental results for cooperative system applied to the simulated robot task in the relatively unconstrained environment.

The "trails" of the circular robot are a trace of its movement after the cooperative GA/NN/LCS/Q-learning system has evolved an effective behavior.

In the relatively constrained environment, the cooperative system demonstrates similar performance, as shown in Figure 3. An effective behavior develops in approximately 1000 time steps. The GA continued refining the network to a final state with approximately 42 hidden layer neurons and 28 percent connectivity.

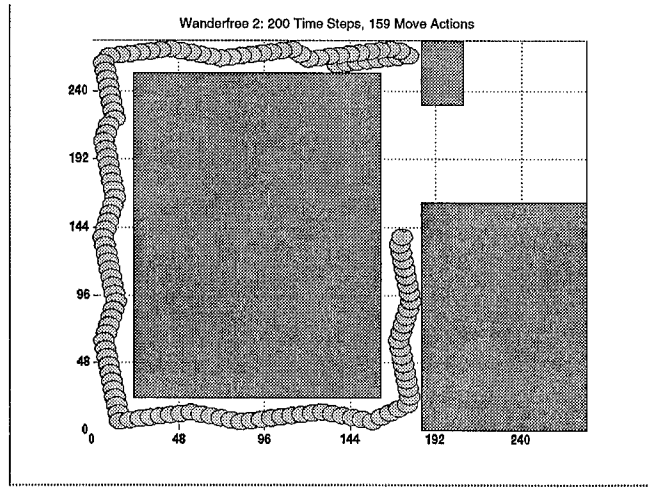


Figure 3. Experimental results for cooperative system applied to the simulated robot task in the relatively constrained environment.

Comparison experiments with random cooperative networks have been performed. These networks used numbers of hidden layer neurons and percentages of input-to-hidden layer connectivity that were similar to the final results given by the evolved system. Such networks fail to learn a set of Q-values that yield organized, effective robot

behavior. This illustrates that the evolved network is effectively organized by the GA for parsimonious Q value storage in the given task.

Examinations of the firing patterns of the hidden layer neurons in the evolved system reveal that these neurons do cooperate to suggest Q-values for each action. This cooperativity is an artifact that has been observed in previous work with supervised neural LCSs (Cribbs, 1995; Smith & Cribbs, 1994). Before examining the nature of these cooperative effects, it is useful to examine similar experiments with a competitive system

5. Competitive System Experiments

Given positive results with the cooperative system, the following sections examine a similar competitive system.

5.1 Experimental Details

Details of the competitive system experiments are largely the same as those of the cooperative system discussed above. However, in this case a conflict resolution system, much like that employed in a LCS, is used to pick a single neuron in the hidden layer at each time step as the "winner". The winner's output alone is used to compute Q-values at the output layer. Competition is based on the maximum output weight for each neuron, after noise is added. The selection of the winner, therefore, is based on the active neuron with the highest noisy Q-value. Note that this is equivalent to selecting in a noisy auction based on strength in the LCS. The other neurons are filtered out and only the winning neuron is allowed to pass its action bids to the output neurons.

In the competitive system, it was necessary to adjust the noise parameter and encourage more exploration early in the run. This is because actions for the system are based on one neuron. If one neuron obtains high Q-values early in the run, it will prevent other neurons from being updated. This is not as much of a concern in the cooperative system, where all neurons are updated whenever they are active. This being the case, it was desirable to increase the noise level early in the competitive run. However, it was also desirable to compare the cooperative and competitive systems at the same noise level late in the run. Therefore, a noise switching operation was added to the competitive system. The system starts with noise level of 1000, and at 25000 time steps, it reduces the noise level to 20.

5.2 Other Experimental Details

The competitive system fitness measure is similar to that used in the cooperative system. However, the competitive system defines the FIRE variable in the fitness calculation as the one for the winning neuron, and zero for all others.

Parameters for the experiments presented here are given in Table 2.

Parameter	Value
Q-learning	
α	0.2
γ	0.75
Noise Amplitude	1000, 20
Neural Network	
Initial Learning Rate	0.016
Learning Rate Increase Rate	1.05
Learning Rate Decrease Rate	0.7
Learning Rate Adaptation Threshold	1.04
Averaging Window	250
Genetic Algorithm	
Block Size	10
p_c	0.8
p_m	0.01
$p_{\#}$	0.1
Population Size	100
Trigger	15

Table 2. Experimental parameters for the competitive system.

5.3 Competitive Results

In typical experiments with the suggested competitive system and the relatively unconstrained environment, the robot eventually learned an effective behavior (see Figure 4). However, it took approximately 30000 time steps for this behavior to emerge. At the end of the run, the neural network had 11 co-evolved hidden layer neurons, and approximately 37 percent connectivity between the input and hidden layers.

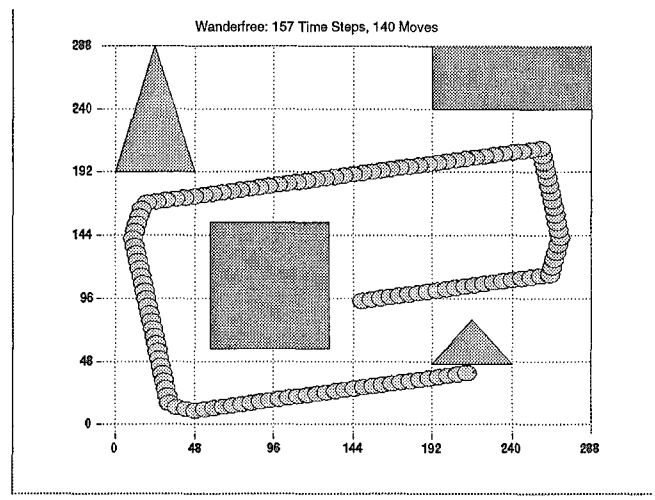


Figure 4. Experimental results for competitive system applied to the simulated robot task in the relatively unconstrained environment.

In the more constrained environment, the competitive system fails to evolve an effective behavior. The systems evolution stagnates, and the robot can only move a few steps before colliding with a wall (Figure 5). After over 60000 time steps, and 100 GA generations, the final network contains 5 hidden layer nodes and approximately 40 percent connectivity.

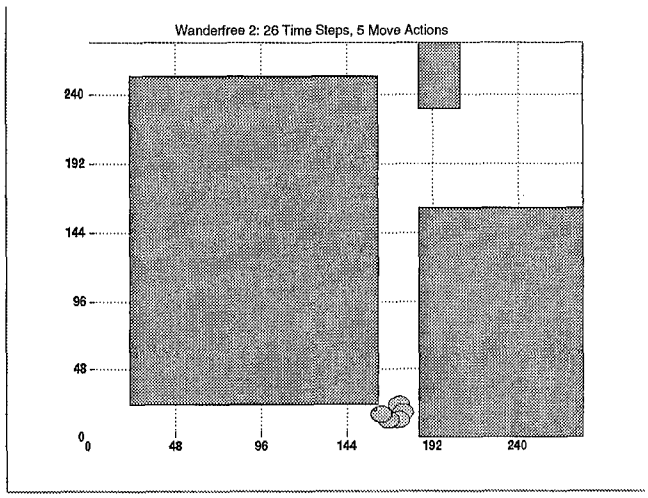


Figure 5. Experimental results for competitive system applied to the simulated robot task in the relatively constrained environment.

The main conjecture of this paper is that the competitive system's slow evolution and relative failure when compared to the cooperative system is due to a failure to co-evolve effective rule sets, as is discussed in the following section.

6. Cooperative-Competitive Comparisons

To be successful in the simulated robot tasks presented here, elements of the control system (whether competitive or cooperative) must evolve into fairly simple, symbiotic relationships. Simply stated, the system must say: IF an object is sensed immediately ahead THEN turn, ELSE move forward.

Note that, due to the structure of the systems involved, the "THEN" and "ELSE" portions of this rule must be implemented by different, coevolved system elements (i.e., hidden layer neurons or classifier neurons). To view this as a two-element operation is something of a simplification, since the concept "an object is sensed immediately ahead" actually represents a variety of sensor states. However, one can view the overall IF-THEN-ELSE relationship involved in this strategy as one of the simplest and most necessary forms of coevolved behavior: a *default hierarchy* (Holland, et al., 1986). The symbiotic relationships involved in default hierarchies are clear:

- one or more elements of a coevolved system act as exceptions that overrule the actions of one or more elements that act as defaults.

The success of the cooperative system is due to the emergence of default hierarchies. The relative failure of the competitive system is due to the fact the such hierarchies fail to sustain themselves in several important instances.

In the relatively unconstrained environment, the default hierarchy necessary is simpler than in the relatively constrained environment. In the unconstrained environment, the system can have an effective behavior simply by turning in one direction when an object is directly ahead. In the relatively constrained environment, the simulated robot must have both left-hand and right-hand turning rules to negotiate the narrow hallway.

Although default hierarchies are an often-discussed, key issue in competitive LCSs, difficulty in their formation and maintenance is not uncommon (Smith, 1991). In our cooperative experiments, symbiosis seems to form more readily. It is important to note that some default hierarchical relationships do form in the competitive system presented here. Also, it may be possible to change experimental parameters or details of the competitive system presented here and achieve more success through better default hierarchy formation. However, our results (on these experiments, and many others) indicate that default hierarchy formation may be easier in cooperative systems. These observations are only anecdotal at this time, but they seem consistent throughout our experiments. Examination of the emergent interactions in the systems seems to reveal the reasons for this effect.

In the cooperative system an element that implements a default action can form and act, receiving its appropriate reward or Q-value update. Later, an exception can emerge through the action of the GA, and *supplement* the activity of the existing elements. The existing elements continue to act in all the contexts in which they previously acted. The new elements only supplement and modify the cooperatively formed actions. Our results clearly show this effect in the cooperative systems. Although detailed analysis cannot be included here for the sake of brevity, the results show that default rules are overridden by exception rules. However, even when the exceptions act, the default continues to act and receive appropriate Q-value updates.

In the competitive system, new elements (exceptions) act entirely separately from the elements to which they are symbiotically linked (defaults in a hierarchy). The exception must override the effects of the default through beating it in the competition for action formation. This prevents the default from receiving a Q-value update, which would appropriately reflect its value as a participant in the default

hierarchy. In our system, this apparently causes the GA to often delete one participant in the default hierarchy prematurely. The symbiotic effects of the default on the exception's value (and vice versa) are stretched out in time. This temporal element complicates the coupling of Q-value updates and GA evolution.

One might try to overcome this difficulty in a competitive system by assigning some update to elements that do not win (and, therefore, formulate the current action), but do match the current state. This would assume that they must be participants in a symbiotic structure like the default hierarchy. However, early in the run, they are more likely to be useless elements that simply are in the right place at the right time. Such elements, if updated, could become parasites (Smith, 1991) that harm overall system behavior.

It is useful to note that the cooperative system may also have positive effects on the development of elements that act symbiotically through *chains*. The symbiotic relationship involved in chains is as follows:

- one or more elements of a coevolved system act to formulate an action that positively influences the actions formulated by other elements at a later time.

The temporal credit assignment algorithm (in this case, Q-learning) acts to assign appropriate credit in chains. However, it takes time for credit to propagate through a chain, regardless of the credit assignment algorithm employed. We theorize that since elements can sustain themselves in a symbiotic team more easily in the cooperative system, they can persist longer than in the competitive system, and allow for appropriate temporal credit assignment to take place.

7. Final Comments and Future Directions

Our experimental results lead us to believe that certain types of symbiotic relationships can evolve more easily between elements that cooperate to form behaviors than between elements that compete for the right to behave. In a cooperative scheme, all elements that respond to a given state are updated as a symbiotic team. In a competitive scheme, they must compete to be evaluated. This slows and complicates appropriate evaluation of new elements under the competitive scheme. In default hierarchies, exception elements must actually win competitions with existing default elements (to which they should be symbiotically linked) to be evaluated. In chains, elements must sustain themselves sufficiently to win competitions and get appropriately rewarded through temporal credit assignment. In both cases, the symbiotic relationships in a competitive system are stretched out in time, which complicates fitness evaluation, and the effects of evolution. We are continuing experiments in more complex settings to confirm these observations.

Although the experiments presented here are drawn from AI, we believe the results represent an interesting area for examination in natural and other adaptive systems. The investigation presented here forms a foundation for examining the roles that behavioral competition and cooperation play in coevolution.

There are also interesting future directions for examination of the system presented in this paper, which include:

- Mixed competitive-cooperative approaches.
- Refinement of exploration/exploitation strategies.
- Examination of reinforcement learning schemes other than Q-learning.
- Allowing the GA to manipulate more extensive structures in the network (e.g., neuron groups, perhaps with more complex functionality and recurrence).
- Development of a suite of test environments and appropriate statistics to examine coevolution in a more rigorous way.
- More complex and realistic robot simulations, leading towards implementation in an actual robot, or groups of robots.

Finally, the system presented offers an interesting, general platform for the further examination of co-evolving systems. If the effects of interactions (like cooperative and competitive action formulation) on underlying evolution can be better understood, the utility of adaptive systems (in both prescriptive and descriptive settings) can be enhanced.

Acknowledgments

The authors acknowledge hours of useful discussion and pages of email with Lashon Booker and Stewart Wilson that greatly influenced this work. The authors gratefully acknowledge support provided by the National Science Foundation (grant ECS-9212066).

References

Barto, A. G. (1990). Some learning tasks from a control perspective. COINS Technical Report No. 90-122. University of Massachusetts. Amherst, MA.

Barto, A. G. and Bradtke, S. J. and Singh, S. P. (1991). Real-time learning and control using asynchronous dynamic programming. COINS Technical Report No. 91-57. University of Massachusetts. Amherst, MA.

Carpenter, G. A. and Grossberg, S (1988). The ART of adaptive pattern recognition by a self-organizing neural network. *Computer,* 21(2), 77-88.

Cribbs, H. B. (1995). Cooperative learning classifier systems: A network approach. TCGA Report No. 95002. University of Alabama. Tuscaloosa. (Master's Thesis).

Goldberg, D. E. (1989). *Genetic algorithms in search, optimization, and machine learning.* Addison--Wesley. Reading, MA.

Hecht-Nielson, R. (1991). *Neurocomputing.* Addison-Wesley. Reading, MA.

Holland, J. H., Holyoak, K. J., Nisbett, R. E., and Thagard, P. R. (1986) *Induction: Processes of inference, learning, and discovery.* MIT Press. Cambridge, MA.

Holland, J. H. and Reitman, J. S. (1978). Cognitive systems based on adaptive algorithms. In Waterman, D. A. and Hayes-Roth, F. (Eds.) *Pattern Directed Inference Systems.* Academic Press. New York.

Kohonnen, T (1984). *Self-organization and associative memory.* Springer Series in Information Sciences 8. Springer-Verlag. New York.

Mahadevan, S. and Connell, J. (1992). Automatic programming of behavior-based robots using reinforcement learning. *Artificial Intelligence,* 55, 311-365.

Rummery, G. A. and Niranjan, M. (1994). On-line Q-Learning using connectionist systems. CUED/F-INFENG/TR 166. Cambridge University.

Smith, R. E. (1991) Default Hierarchy Formation and Memory Exploitation in Learning Classifier Systems. TCGA Report 91003. Tuscaloosa, AL. (Ph.D. dissertation, University Microfilms No. 91-30,265).

Smith, R. E. and Cribbs, H. B. (1994). Is a classifier system a type of neural network? *Evolutionary Computation,* 2(1), 19-36.

Smith, R. E. and Cribbs, H. B. (in press) Parsimonious Neural, Genetics-Based Q-Learning
for Autonomous Systems.

Thrun, S. B (1992). The Role of Exploration in Learning Control. In White, D. A. and Sofge, D. A. (Eds.) *Handbook of intelligent control: Neural, fuzzy, and adaptive approaches* (pp. 527-560). Van Nostrand Reinhold. New York.

Twardowski, K. (1993). Credit assignment for pole balancing with learning classifier systems. In S. Forrest (Ed.), *Proceedings of the Fifth International Conference on Genetic Algorithms* (pp. 238-245). Morgan Kaufmann. San Mateo, CA.

Vogl, T. P., Mangis, J. K., Rigler, A. K., Zink, W. T. and Alkon, D. L. (1988). Accelerating the convergence of the backpropagation method. *Biological Cybernetics,* 59, 257-263.

Watkins, J. C. H. (1989). Learning with delayed rewards. Unpublished doctoral dissertation. King's College, London.

Wilson, S. W. (1985). Knowledge growth in an artificial animal. In Grefenstette, J (Ed.), *Proceedings of an International Conference on Genetic Algorithms and Their Applications* (pp. 16--23).

Wilson, S. W. (1990). Perceptron redux: Emergence of structure. In S. Forrest (Ed.), *Emergent Computation: Proceedings of the Ninth Annual Conference of the Center for Nonlinear Studies on Self-organizing, Collective, and Cooperative Phenomena in Natural and Artificial Computing Networks* (pp. 249-256). Amsterdam: North-Holland.

Wilson, S. W. (1994). ZCS: A zeroth level classifier system. *Evolutionary Computation,* 2(1), 1-18.

Wilson, S. W. (1995). Classifier fitness based on accuracy. *Evolutionary Computation,* 3(1), 149-176.

Co-evolution of Pursuit and Evasion II: Simulation Methods and Results

Dave Cliff[1] and Geoffrey F. Miller[2]
[1]School of Cognitive and Computing Sciences
University of Sussex, BRIGHTON BN1 9QH, U.K.
davec@cogs.susx.ac.uk
[2]Center for Adaptive Behavior and Cognition
Max Planck Institute for Psychological Research
24 Leopoldstrasse, 80802 MUNICH, Germany.
Miller@mpipf-muenchen.mpg.de

Abstract

In a previous SAB paper [10], we presented the scientific rationale for simulating the co-evolution of pursuit and evasion strategies. Here, we present an overview of our simulation methods and some results. Our most notable results are as follows. First, co-evolution works to produce good pursuers and good evaders through a pure bootstrapping process, but both types are rather specially adapted to their opponents' current counter-strategies. Second, eyes and brains can also co-evolve within each simulated species – for example, pursuers usually evolved eyes on the front of their bodies (like cheetahs), while evaders usually evolved eyes pointing sideways or even backwards (like gazelles). Third, both kinds of coevolution are promoted by allowing spatially distributed populations, gene duplication, and an explicitly spatial morphogenesis program for eyes and brains that allows bilateral symmetry. The paper concludes by discussing some possible applications of simulated pursuit-evasion coevolution in biology and entertainment.

1 Introduction

At SAB94 we argued that it's important, interesting, and useful to simulate the co-evolution of pursuit and evasion strategies, and presented some preliminary results [10]. In that paper we presented a review of the relevant literature (e.g. [7]); since then, other comparable work has appeared (e.g. [13, 14]).

In the intervening two years of project development, we have improved our methods and generated more detailed and more complete results. This paper gives an overview of our current methods and some results from this ongoing research. We include only the briefest recap of the intellectual motivations and scientific background for this project, which we have covered at length elsewhere [9, 10, 11, 6]. Much of our effort in this project was directed at improving our simulation methods to cope with the sometimes frustratingly slow and baffling process of coevolution. The populations were changed from traditional "pools" to spatially organized matrices with local competition, mating, and replacement. Many evaluation functions were tried and failed before a few were found that worked. We now use a more efficient contest method where individuals play the "Last Elite Opponent" (LEO) [14, 12]. New analysis methods were developed to see whether directional progress is occurring during coevolution, including cross-generational bit-string correlation matrices and gene persistence plots (see [6]). This paper is offered in part as a traveler's advisory about the pitfalls of venturing unprepared into the Red Queen's protean world.

In brief, the intellectual background for our work stems from the observation that animals survive and reproduce by exploiting "fitness affordances" [11] in their environment, and that some of these affordances are themselves mobile animals that do not want to be exploited (e.g. prey, coy females, hosts avoiding parasites). This basic conflict of interest between two mobile animals typically leads to a pursuit-evasion contest, where one animal tries to catch the other to do something to it (eat it, mate with it, suck its blood). Over generations, these contests result in a co-evolutionary arms race between pursuit strategies and evasion strategies, where animals evolve to be faster, more maneuverable, better at predicting each other's next moves, and better at being unpredictable [10, 9]. Pursuit and evasion have been studied from different angles by behavioral biology, neuroethology, differential game theory, and previous SAB work. Thus, pursuit-evasion contests are among the most important, challenging, and co-evolutionary of all animate behaviors. But in their scientific analysis, there is a large gap between the over-simplicity of game-theory models and the baffling complexity of real pursuit-evasion wetware as studied by neuroethology. SAB-style simulation may help fill the gap, by illuminating the co-evolution of strategies that are complex enough to include interesting examples of perceptual specializations, prediction, and proteanism, but not too complex to analyze.

2 Simulation Methods

Our pursuit-evasion coevolution simulator has been developed and refined to serve the study of co-evolutionary arms races, to characterize the resulting behavioral strategies, and to analyze the underlying sensory-motor architectures that generate the observable behaviors through interaction with the environment. The simulation approximates a spatiotemporally continuous system by updating a model of the animats in their environment at a rate of 100 frames per simulated second. Here we summarize some of the most significant details.

The following sections describe the equations governing the physical dynamics of motion for the animats, the artificial neurons used in the simulations, the genetic encoding of the controller specifications as bit-strings for operation on by the genetic algorithm, and then details of the genetic algorithm itself.

2.1 Physical Dynamics

In all our work to date, we have employed a two-dimensional (2-D) simulation, where circular animats chase each other around on an empty infinite plane: there are no obstacles or boundaries in the animats' world. The primary motivation for such a gross simplifying assumption is one of computational economy. Despite the lack of a third spatial dimension, we employed a fairly realistic model physics.

The neural network for each animat gives two output values referred to as v_l and v_r: these are treated respectively as left and right motor signals, giving a differential-steer system (i.e. the rate of change of angle of orientation is dependent on the difference between the left and right motor settings, as in many two-drive-wheeled mobile robots). The equations of motion are based on simple Newtonian point physics.

Specifically, the relevant equations for translating from the output values to the accelerations are:

$$m\frac{d^2f}{dt^2} + c_f\frac{df}{dt} = k_f\frac{v_l + v_r}{2} \qquad (1)$$

for forward movement, where m is the animat mass, c_f is a friction coefficient, and k_f is a scale factor; and:

$$\vartheta\frac{d^2\alpha}{dt^2} + c_a\frac{d\alpha}{dt} = k_a(v_l - v_r) \qquad (2)$$

for changes of orientation angle α, where ϑ is the animat moment of inertia, c_a is an angular friction coefficient, and k_a is a scale factor.

The use of these 'realistic' equations of motion mean that the animats take time to accelerate to a desired speed (either linear or angular). Note also that there are no 'brakes': rather, the drag/friction coefficients mean that if there is a reduction in motor output, the speed falls exponentially towards the new steady-state value.

The values of $m, c_f, k_f, \vartheta, c_a$, and k_a are held fixed for any one experiment, but we have experimented with a number of settings of these parameters. Clearly, the relative settings can have a large effect on what constitutes a sensible strategy for pursuit or evasion. In the limit, the relative settings can determine the likelihood of anything interesting happening. Figure 1 shows an impressionistic rendering of the effects the relative pursuer/evader dynamics have on the outcome of the contests. A schematic 2-dimensional space is shown, with the horizontal axis being a measure of the angular motion capabilities of the pursuer (P) relative to the evader (E), and the vertical axis measuring the relative linear motion capabilities. In the horizontal, there will be some threshold value past which the pursuer can reliably out-turn the evader, and in the vertical there will be a threshold beyond which the pursuer can in principle catch up with the evader. These two threshold lines partition the space into four zones. The further into the upper-right zone, the greater the chances of pursuers winning all contests (they can catch up with the evaders, and out turn them too); the further to the lower-left, the more likely the evaders are to do very well (the pursuers can neither catch them nor out-turn them). Thus, if there are asymmetries in the dynamics (i.e. the pursuers and evaders have different parameters for the equations of motion) then for interesting trials, the pursuer's maximum linear speed should be greater than the evader's (so the pursuer can, in principle, catch the evader), and the evader's maximum angular speed should be greater than the pursuer's (so the evader can, in principle, dodge and out-turn the pursuer).

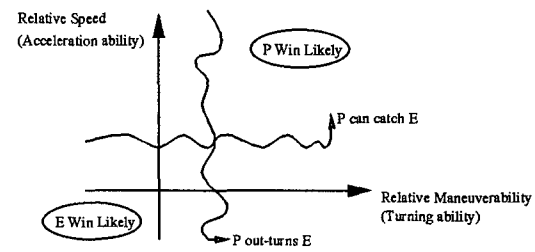


Figure 1: Impressionistic illustration of the effects of relative kinematics on the likely outcome of a pursuit-evasion contest. See text for explanation

The results described later in this paper come from experiments where the kinematics were *symmetric*: that is, the pursuers and evaders had identical parameter values for their equations of motion. Thus, the pursuer and evader have the same accelerations and maximum speeds, which gives a clear advantage to the evader. This is because, as long as the trial starts with sufficient distance between the pursuer and evader for the evader to turn to face away from the pursuer before the pursuer

can hit the evader, the evader need only perform such a turn and then accelerate to top speed to avoid being caught – its top speed is the same as the pursuer's. To redress the balance, the animats were given a limited stock of 'energy' or 'fuel', with the pursuer having (slightly) more initial energy than the evader. Energy consumption rose as a quadratic function of linear force exerted, so an animat could 'dawdle' by moving slowly for a long period of time, or 'burn out' by accelerating to maximum speed and using up all its energy in a few seconds. When energy fell to zero, the motor outputs were disabled and the animat would drift to a halt, being slowed by the simulated friction. Small amounts of random noise were also added onto the values of v_l and v_r, so that movement was nondeterministic.

2.2 The Neuron Model

We used continuous-time recurrent neural networks of a type which have recently been the subject of detailed analysis (e.g. [1]). The activity of a single 'neuron' processing unit is described by the equation:

$$\tau_i \frac{dy_i}{dt} = -y_i + \sum_{\forall j} w_{ji}\sigma_j(y_j) + \mathcal{N}_i(t) + \mathcal{I}_i(t) \qquad (3)$$

Where: y_i is the activity of unit i; τ_i is the time-constant for unit i; $\sigma_j(\xi)$ is a sigmoidal function with bias term (threshold) θ_j; w_{ji} is the weight of the connection from unit j to unit i; $\mathcal{N}_i(t)$ is spontaneous noise injected at time t, from a uniform distribution centered on zero; and $\mathcal{I}_i(t)$ is external (sensory) input at time t.

For each unit, the values for θ_i and τ_i, and the upper and lower bounds on the distribution of $\mathcal{N}_i$ explicitly set by a sequence of bits on the genotype. The values for w_{ji} and $\mathcal{I}_i$ were also genetically specified, but not explicitly. Rather, they were derived from the 'morphogenesis' process: this is described in Section 2.3. The connection strengths w_{ji} remained constant throughout the 'lifetime' of each animat: in this sense, there was no opportunity for 'learning'.

We had only one sensory modality: simulated 'flatland vision' in the 2D plane. Photoreceptor cells could be placed on the perimeter of the animat's circular body, with their location, orientation relative to the body, and angle of acceptance all being genetically specified. The response of a photoreceptor was proportional to the percentage of its acceptance angle that was *not* occupied by the opponent's circular body. The relevant 2-D projection equations are trivial when it is known that there is only one object in the visual world. The equations could thus be solved analytically, without recourse to numeric approximation techniques; but random noise was added in to ensure that sensing was nondeterministic and that there was a lower limit on resolution (i.e. small signals could be swamped by noise).

2.3 Genetic Encoding and Morphogenesis

The architectures for the neural-network sensory-motor controllers were specified by bit-string genotypes. In common with past work in evolving such controllers for artificial autonomous agents (e.g. [4]) we treat the arrangement of the sensory morphology as an intrinsic part of the controller-network specification. The controller-network is encoded as strings of binary digits (bits). Rather than use variable-length genotypes to allow varying numbers of units in the networks (e.g. [4]), we use fixed-length genotypes, partitioned into a number of 'fields'. At the start of each field is a sequence of bits which governs whether that field is 'expressed' – i.e. whether it is ignored or read to form part of the specification for the network. Each field contains bit-sequences that govern parameters that define an individual neuron: the neuron has a particular genetically-specified physical location within the animat's 2D circular body; and there are genetically set parameters that determine the growth (on the body) of a fractal 'input' tree and a fractal 'output' tree. If the input tree of one unit intersects the output tree of another unit, and certain other genetically specified conditions are met, then a connection is made joining the input to the output, with the strength of the connection being affected by the geometry of the intersection of the two trees. If the unit's output tree terminates within a central zone on the animat's body, then it becomes a motor output unit, affecting the left or right output value depending on the precise geometry of the termination point(s). If a unit's input tree extends beyond the perimeter of the body, then that unit becomes a visual input unit, with the location, orientation, and acceptance-angle of the corresponding photosensor being set by the geometry of the tree's intersection with the body-edge. Finally, each field contains a sequence of bits that can specify *symmetric* expression: if a neuron is expressed symmetrically, then a copy is created by reflecting the neuron's body and input and output trees about the animat's longitudinal axis. Full details of the encoding scheme are given in [3].

This encoding scheme has proven to offer considerable power. It allows for some units in the network to be "hidden"(cf. interneurons), others to be sensory input neurons, and others to be motor output neurons. But most interestingly, it is possible for a unit to be *both* sensory *and* motor. This capability, coupled with the relative ease of generating bilaterally symmetric designs, means that random genotypes often encode simple but very effective Braitenberg-*Vehicle*-like controller architectures (cf. [2]). Almost equally as often, random designs may have no motor outputs, no sensory inputs, or no connection between sensors and motors.

For this reason, we use "animat eugenics" to increase efficiency: when generating the initial random population of genotypes, or breeding a new individual from two

parents, a viability check is made: individuals with insufficient numbers of sensors or motors are rejected immediately. Then the genome is expressed to give a network, and a garbage collector deletes any disconnected units. If, after garbage collection, the network is still viable, it is added to the population. Otherwise, the genotype is discarded and another new one generated and tested: the process repeats until a viable genotype is generated.

One aspect of using this encoding scheme in practice is that typically there are significant sequences of bits on the genotype which constitute 'junk DNA', in that they don't contribute to the phenotype. Bits in a field which has the expression sequence set to "off" have no effect on the phenotype, and nor do bits that specify neurons that *are* expressed but subsequently deleted by the garbage collector because they don't connect to anything. However, we are cautious about the use of the term 'junk DNA': subsequent recombination or mutation could lead to an unexpressed field being expressed or to a disconnected neuron being connected. Thus, we prefer use of terms such as 'silent' or 'resting' DNA rather than 'junk'. One practical effect of silent DNA in the genomes is that a mutation in a silent sequence has no immediate effect. To counter this, we use mutation rates which would be considered very high in more standard encodings where every bit counts.

2.4 The Genetic Algorithm

Our genetic algorithm (GA) involves a population of pursuers and a population of evaders. Both populations are the same size. We use a spatially distributed GA, where individual genotypes in each population are spread out to occupy discrete positions on a grid. Individuals will only breed with nearby neighbors, and the offspring will replace nearby less-fit individuals. The grid has a toroidal topology so there are no edge effects.

The GA proceeds in discrete generations. In the very first generation, each individual is tested against a set of randomly-chosen individuals from the opponent generation. In all subsequent generations, individuals are tested against the best individual in the opponent generation on the previous generation – this "last elite opponent" (LEO) evaluation was introduced in [14, 12].

In each test, the genotype is given a number of *trials*: these are individual pursuit-evasion contests. The initial conditions for the trials are varied using standard statistical precautions. The final fitness of the individual was calculated as the mean fitness score from the set of trials. The evader fitness score was simply the length of time it lasted before being hit by the pursuer. The pursuer's fitness score was slightly more complex: they received fitness points if they were approaching the evader, and received a bonus if they hit the evader; this bonus was dependent on the time of the collision, and was two orders of magnitude bigger than the points received for approaching the evader, so pursuers that collided with evaders were much more fit than those which stealthily tracked their opponents. It should be noted that *many* fitness functions were tried before we arrived at this successful combination.

In each trial, the pursuer and evader start at their initial positions and orientations, with all activities in their neural networks set to zero. Once the trial has started, it continues until the animats collide, or both run out of energy and drift to a halt, or a fixed time-limit is reached.

Once all individuals in both populations have been evaluated, two new populations are bred. The breeding uses the standard GA operators of mutation and crossover, plus a duplication operator discussed further in Section 3.1. To breed a new individual, two 'parent' individuals are chosen using probabalistic rank-based selection. One of the two parents is chosen at random, and the process of copying its genotype into the 'child"s genotype commences. Once each bit is copied, a random number is generated from a uniform distribution. If this is less than some threshold value, then the copying pointer is switched to the corresponding position on the other parent. The number of crossovers in the reproduction of any one 'child' thus follows a poisson distribution: there will always be a finite probability that reproduction is asexual (i.e. number of crossovers is zero). Mutation is also applied on a bit-wise basis, with a different threshold value, so the number of mutations per reproduction also follows a poisson distribution.

3 Results

As is common in our experience of using artificial evolution, we witnessed many failed experiments before we experienced consistent evolution of animats exhibiting desired types of behaviors. Here we make a general observation about what we learned from one of the causes of our many failures, and then give a behavioral-level overview of a run that yielded some interesting results.

3.1 A Lesson from Failures: Duplication

The genetic encoding and morphogenesis technique we employed, along with the viability checks and garbage collection, almost guaranteed that at the start of a run, the most successful individuals would have very simple neural architectures similar to Braitenberg's *Vehicles*; with a symmetrical arrangement of a few (typically two or four) sensors connected to the motors by either 'crossed' or 'uncrossed' connections (i.e. connected contralaterally or ipsilaterally). Very often, any subsequent improvement in performance was due solely to variation and selection operating on the parameters of the individual neurons in the animat: the number of neurons almost never increased, even when there was a capacity

on the genotype for ten or fifteen more. The improvement in performance, even after 1000 generations, was often only a small percentage of the initial performance of the random-genotype architecture.

We came to understand that this was because these small Braitenberg-style controller networks typically have very high epistatic boundaries around them in genotype space. That is, adding in one or more extra neurons *at random* almost always resulted in significant degradation of performance. The initial networks were so small that adding new neurons typically has a huge negative impact on the network's performance.

To remedy this, we borrowed another genetic operator from nature. Whereas many artificial evolution systems employ crossover and mutation, *duplication* is somewhat less widely used. In our implementation, duplication would take a single genetic field and copy it into another field on the genotype. If the field had its expression bits set to an active sequence, the resulting controller network would have extra neurons, but they would be (almost) identical to the units in the current successful network, and so would be unlikely to have a negative effect on the overall behavior. Subsequent mutations could smoothly alter the parameters encoded on one of the copied neuron fields, allowing for the other copy to either remain the same or, more importantly, to change and hence co-adapt to the effects of the new unit(s). Once duplication was employed, we found that the number of units would often increase significantly over the course of a run, and this would be reflected in significant increases in performance. The discussion of Figure 10 in the next section illustrates duplication in action.

3.2 A Run That Worked

Results from individual trials between the elite pursuer and elite evader at various stages in a co-evolutionary experiment are shown in Figures 2 to 9. These figures all come from the same single run with pursuer and evader population sizes of 100, arranged in 10×10 grids, lasting for 1000 generations. We used LEO evaluation, with 15 trials per test; fitness being the average score. The time-limit on each trial was 15 simulated seconds, which typically took less than one second of real time. Animats were viable if they had at least two motor output neurons and at least one sensory neuron. The genetic algorithm was configured to give (on the average) 1.1 crossovers, 7.0 mutations, and 0.1 duplications per reproduction event. The genotypes had seven unit-fields, so the maximum number of units in an animat's network is 14 (i.e. all 7 fields expressed and bilaterally symmetric).

Figure 2 shows traces of the 2D paths made by the elite pursuer and evader from generation 0. These two animats are the best of the initial population, with randomly-specified genotypes. Traces of some significant variables are shown in Figure 3: the first nine graphs are, from the top: distance between animats (in meters: the animat bodies have a radius of 9mm); pursuer's 'target bearing' (in radians: a value of zero implies that the evader is straight ahead); evader's 'target bearing' (in radians: a value of zero implies that the pursuer is exactly behind); pursuer forwards velocity ($m \cdot s^{-1}$); pursuer angular velocity ($radians \cdot s^{-1}$); evader forwards velocity; evader angular velocity; pursuer energy (arbitrary units); and evader energy. The next two graphs are output activities of the pursuer's two 'neural' units, followed by six graphs showing outputs of the units in the evader's network (the networks are shown in Figure 10).

These two figures give some indication of the relatively rich behavior that can be produced by randomly connected noisy continuous-time recurrent neural networks. However, the behaviors exhibited by both the pursuer and the evader are somewhat distant from what we desire: the pursuer is supposed to chase the evader, but instead it starts by moving *away* from the evader. Similarly, the evader is supposed to be running away from the pursuer, but it's not: it moves towards the pursuer, and then (around t=4.3s) it starts turning in a tight circles for about five seconds. By this time, the pursuer has (as a result of random activity in its units) turned around and is heading toward the evader. It continues on a straight path, while the evader wanders around some more. The trial ends after the allotted 15 seconds without a collision: notice also that both animats have significant amounts of energy remaining at the end of the trial. Clearly, there is room for improvement here. But even though these two animats are bad, they are not as bad as others in their populations. Because this is the first generation, and the genotypes for the two populations were randomly generated by the same processes, at this stage some predators may run away at top speed from the evader; some evaders may actively seek the predator; and many animats in both populations will simply do nothing at all.

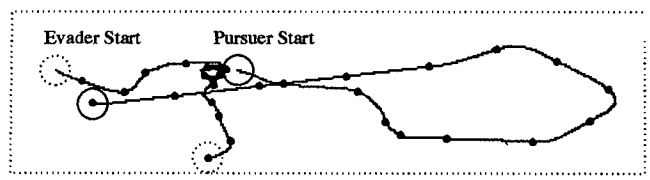


Figure 2: Pursuit-Evasion trajectory from generation 0. Plot shows start position of Pursuer (solid) and Evader (dotted): the circles are the same diameter as the animat 'bodies'. The paths taken by the two animats are marked with dots at one-second intervals, and the positions at the end of the trial are also shown (after 15 seconds). See text for further discussion.

Figure 4 shows the paths from a trial between the elite pursuer and evader after 200 generations of co-evolution. The pursuer is clearly chasing the evader, and the evader

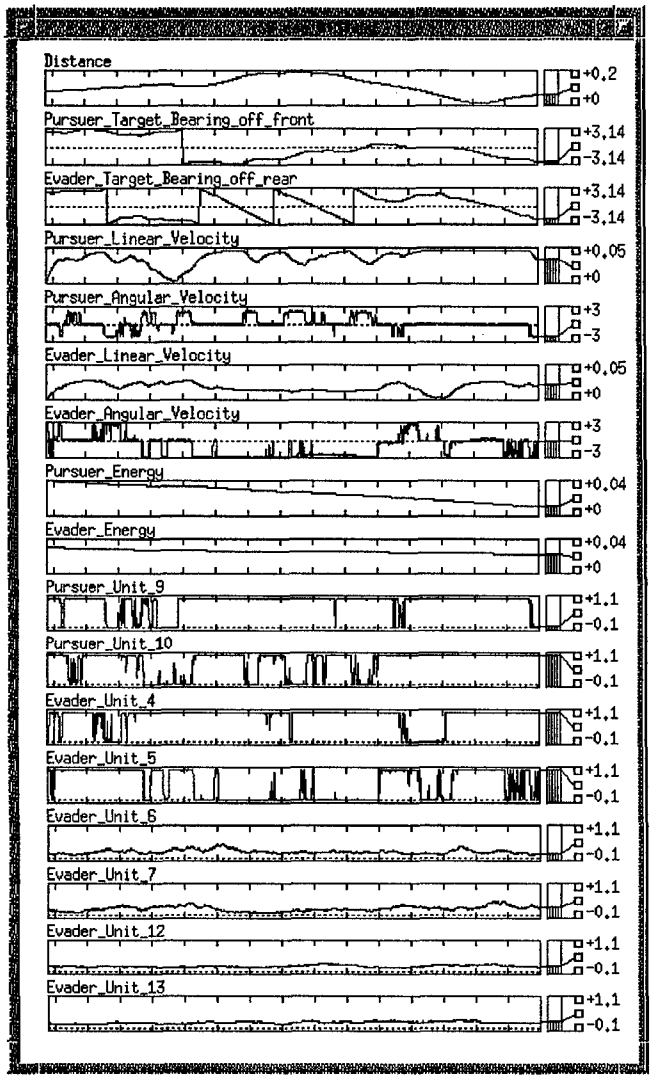


Figure 3: Data traces from the trial shown in Figure 2: horizontal axis is time, with tick-marks at one-second intervals. See text for explanation and discussion.

is clearly running away from the pursuer. But, as is illustrated in Figure 5, the pursuer strategy involves accelerating to near-maximum velocity, slowing only as a consequence of the need to slow one wheel for turning through differential steering. This policy consumes energy rapidly, and at around t=8s the pursuer's energy falls to zero: its motor outputs shut off, and it coasts to a halt. The evader has not been moving so fast, and hence has sufficient energy to last until very near the end of the trial, continuing to move away from the stalled pursuer. So, the pursuer is exhibiting the right kind of behavior but lacks subtlety, and the evader is also showing roughly appropriate behavior, although it is difficult to judge how it would fare against a more sophisticated pursuer.

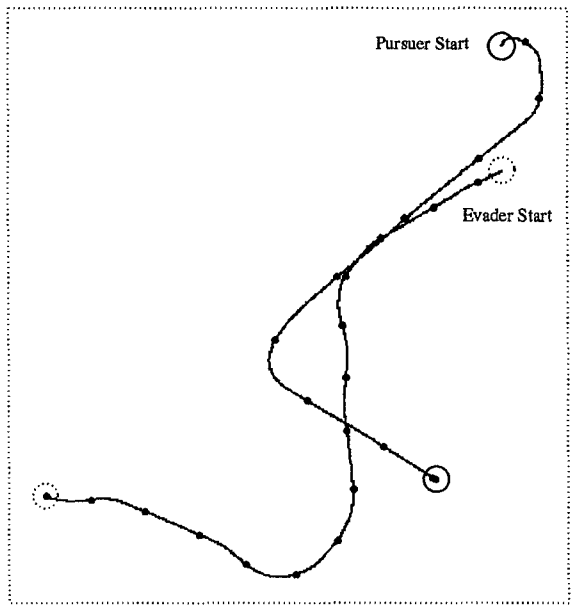


Figure 4: Pursuit-Evasion trajectory from generation 200. Display format as for Figure 2.

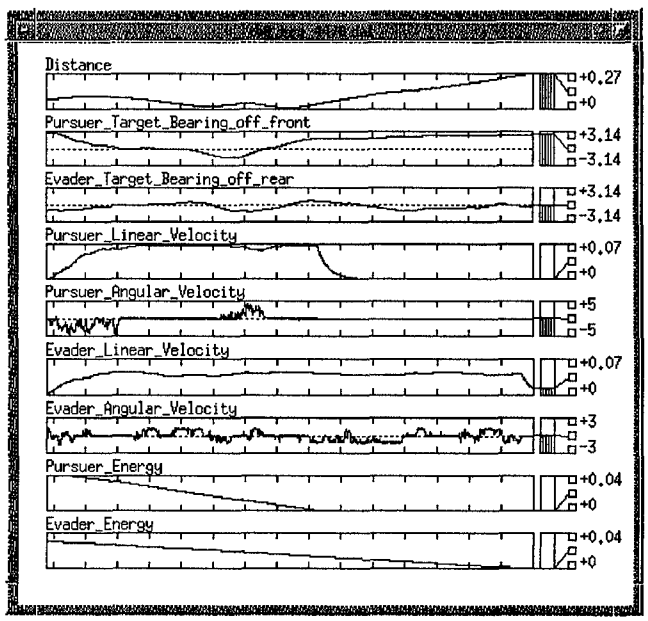


Figure 5: Data traces from the trial shown in Figure 4. Note that the vertical-axis ranges for distance, linear velocity, and angular velocity, have all been increased from those used in Figure 3.

Results from a contest between the elites of the two populations after 999 generations are shown in Figures 6 and 7. As can be seen both from the trajectory and from the graphs of target-bearings for both the pursuer and the evader, the pursuer keeps itself pointed straight at the evader, and the evader keeps the pursuer right on

its "6-o'clock": again the pursuer uses all its energy and drifts to a halt, but its chasing strategy has forced the evader to use much more energy, and only about half a second later the evader also stalls. Presumably, slower-moving evaders have been selected against because they are more readily caught by the pursuers. This is not to say that a slower evader would *necessarily* be worse: a point we illustrate below. Nevertheless, at this stage in the coevolutionary process, neural network controllers for pursuit and evasion have manifestly been successfully co-evolved.

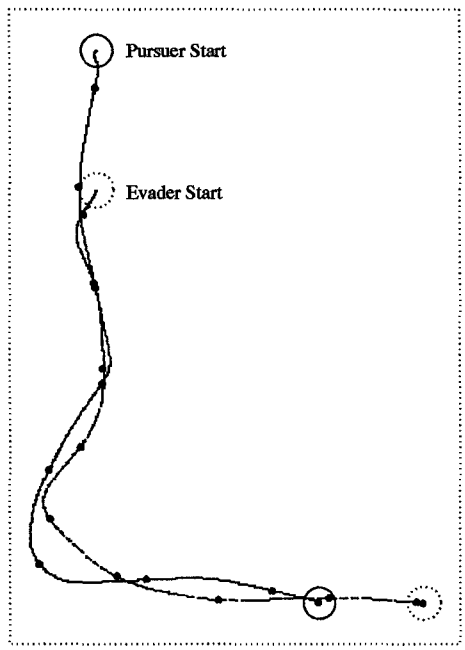


Figure 6: Pursuit-Evasion trajectory from generation 999. Display format as for Figure 2.

So, even after 999 generations, we have a pursuer which fails to catch the evader: this was also the situation at generation 200, and indeed at generation 0. As we discussed at length in an earlier paper [6], to check for progress or improvements in performance, it can be informative to test a current individual against ancestors of its current opponent(s). As an example, Figures 8 and 9 show trajectories and data traces from a trial where the best pursuer from generation 999 is pitted against the best evader from generation 200: as can be seen, the pursuer is indeed capable of catching some types of evaders, but the evaders at its generation are well enough adapted to its pursuit strategy that they get away successfully. It is notable that the generation-999 pursuer is markedly less accurate in chasing the generation-200 evader: it overshoots and corrects a couple of times before the final collision. This may be a sign that the ancestral evader is exhibiting a strategy which the pursuer is *less* well adapted to than the strategy of the evaders at its own

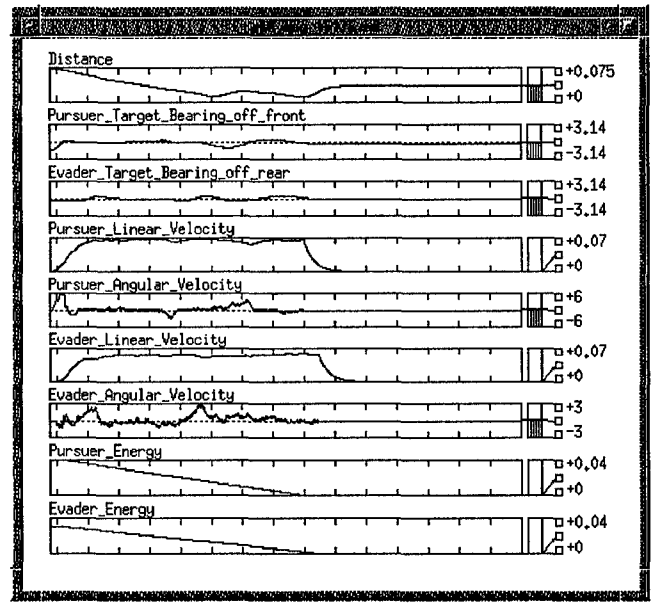


Figure 7: Data traces from the trial shown in Figure 6. Note that, in comparison to Figure 5, the vertical axis range for distance has been decreased, and the axis range for pursuer angular velocity have been increased.

generation: the generation-200 evader holds its speed at approximately 75% of the speed that the generation-999 evader runs at; the generation-999 pursuer may coordinate its turns and lunges with an 'expectation' that the evader is moving at the higher speed, and so it has some trouble with a slow evader from an ancestral generation: this is an example at the behavioral level of the effects of limited "genetic memory" [6].

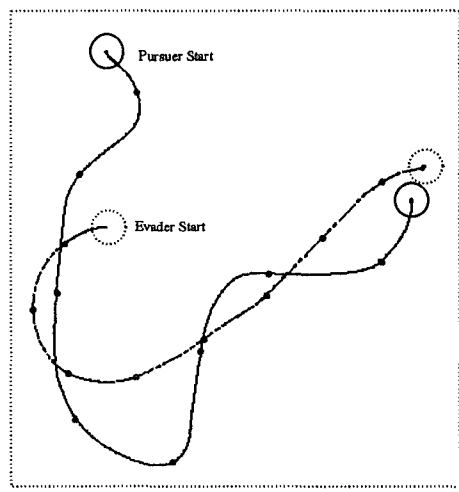


Figure 8: Pursuit-Evasion trajectory resulting from testing evader from generation 200 against pursuer from generation 999. Display format as for Figure 2. Trial ends with collision at t=8.6sec.

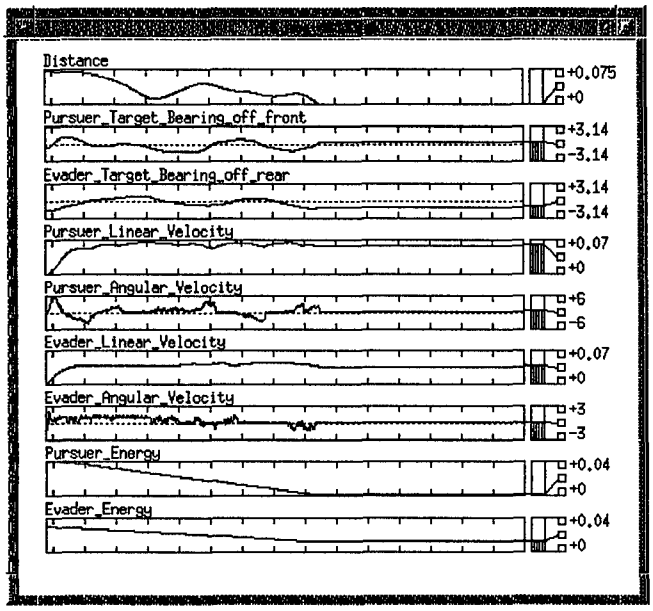


Figure 9: Data traces from the trial shown in Figure 8. Data recording continues after the collision at t=8.6sec, but all values are frozen.

The evolutionary changes in the morphology of the pursuer and sensory-motor networks are illustrated in Figure 10.

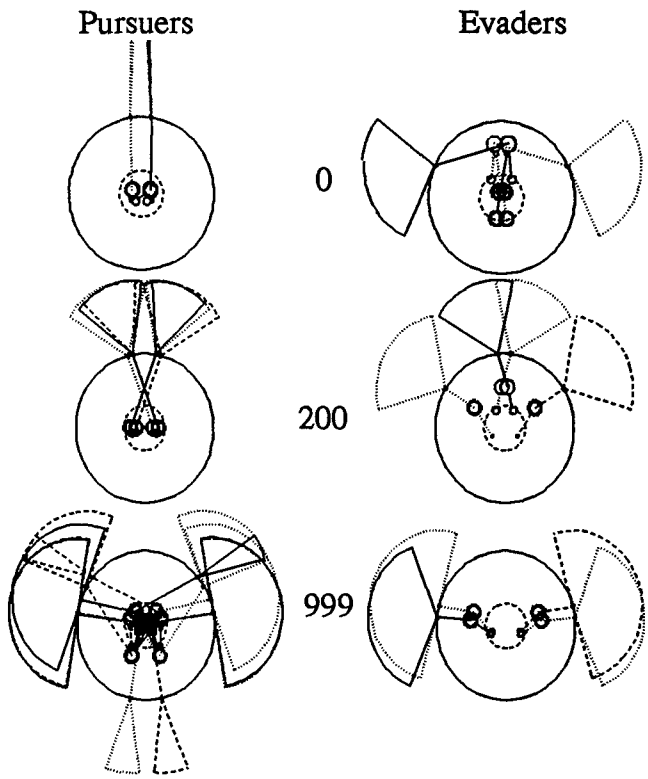


Figure 10: Plots of the sensory-motor morphologies of the elite pursuer and evader at generations 0, 200 and 999. Each plot is a top-down view of the animat, with the front toward the top of the page. The large circle is the extent of the animat's body. The inner dashed concentric circle marks a "spinal chord" zone on the body where the termination of a 'neuron' processing unit's output signifies that the unit contributes to the motor outputs of the animat. Other circles within the animat's body are 'neurons': the larger ones are neuron 'cell bodies' and the smaller ones are 'synapses'; the straight lines are connections joining neurons, via synapses, to each other or to motors or sensors. Visual sensors have genetically specified positions on the periphery of the animat's body, and each has a genetically-specified orientation (i.e. direction of view relative to the body) and angle of acceptance which are indicated here by the circle-segments emanating from the body perimeter. Different line styles are used to help distinguish different units. At generation 0, both the pursuer and the evader have Braitenberg-Vehicle-style architectures, with only two visual sensors each. After 200 generations, the combined effects of duplication and mutation can be seen: it is especially clear that the pursuer's six sensor units are close copies of each other; only a few small mutations have occured since the duplication events. By 1000 generations, the pursuer has a fairly elaborate sensory morphology, with four lateral wide-field photosensors, four frontal narrow-field sensors, and 2 rear-facing narrow-field sensors. In contrast, the evader has four laterally positioned wide-field sensors, without the frontal narrow-field or rear-facing sensors that the pursuer has.

4 Further Work

Our work to date has demonstrated that the co-evolution of continuous-time recurrent neural-network controllers for pursuit and evasion strategies in environments with realistic sensory-motor dynamics is a reasonable thing to do. In this section we discuss some of the future directions this research could be taken in.

4.1 Applications in Biology

As we discussed in [10], there is a wealth of biological literature on pursuit and evasion in animals, with analysis at a variety of levels ranging from evolutionary arguments down to neuroethological studies of neural circuits governing pursuit/attack and evasion/escape.

In [5] it is noted that, just as artificial evolution can be used to semi-automatically propose new designs for the sensory-motor control architectures of artificial autonomous agents, so it could (in principle at least) be used to propose new 'designs' for neural sensory-motor coordination mechanisms in animals: the designs would effectively be models for the neuroarchitecture underlying an observable behavior in an animal. For this to be workable, it would be necessary to set up a simulation system where the physical and neural dynamics of the evolving animats are biologically plausible, and the evolutionary process only generates biologically plausible architectures (or selects very heavily against any implausible architectures): in selecting for animats which exhibit a particular behavior, the evolutionary process could generate a model or models for the neural mechanisms underlying that behavior, which may subsequently drive empirical or theoretical biology: an early example

of such work is [15]. Hence, one possibility is that we use our simulator for studying visually guided pursuit behavior in animals. Such behavior has been studied extensively in the biology literature. One system particularly well understood at the neural level is chasing behavior in house-flies (see e.g. [8]). If we attempt to evolve pursuer animats with fly-like dynamics, which produce fly-like behaviors, we may learn more about real flies.

Before such work can commence, it is likely to be necessary to make the (visual) environment more challenging. Currently, the only object in either animats' sensory world is another animat – i.e. its opponent. This abstracts away much of the difficulty of real visually-guided pursuit and evasion. In real worlds, it can be necessary to distinguish a moving target from a cluttered background (the image of which may also shift due to self-motion), or to deal with predicting the future path of a target to adaptively cope with brief occlusions. The fact that most of our evolved animats suffice with a small number of photoreceptors is probably due to the simplicity of the visual environment, and to evolve visual architectures of interest to biologists, it will probably be necessary to perform experiments in more visually complex worlds.

The most simple means of increasing visual complexity is to add static obstacles into the animat-world. Of more long-term interest (and complexity) is the possibility of having more than two agents in the environment at any one time. Even with only three animats, there are possibilities of exploring the effects of the system's (predetermined) dominance hierarchies: they could be defined transitively (giving a "food-chain" type scenario) or intransitively, forming a cycle; and it becomes possible to study issues in attention. As the number of animats in a trial increases, so it may be possible to study group behaviors such as pack-hunting and coordinated evasion (e.g. random scatter). Such group dynamics could also allow for studying signaling and communication [16].

Furthermore, in many species of prey (i.e. evader) animals, there are often three distinct phases to an encounter with a predator (i.e. pursuer) animal. At first the prey will be vigilant, conserving energy. When the predator comes closer, the prey will engage in linear fleeing, where it moves in a straight line at high speed, away from the predator. Once the predator is within some nearby distance threshold, the prey will then switch to the third phase, involving protean jinking and dodging behavior. Given the need for energy economy in our simulations, and game-theoretic arguments for the optimality of mixed (i.e. random) strategies in a variety of pursuit-evasion scenarios, it would be intriguing to see whether a similar three-phase evader strategy evolves without explicit selection for such phasic strategies. To keep the analysis clear, it would be necessary to work with non-differential-steer animats, so that forwards speed is not necessarily affected by making turns. We would then hope to see a negative correlation between pursuer-distance and evader speed (i.e. a primitive form of vigilance), and a nonlinear relationship between absolute angular speed and pursuer-distance, with a peak near to the threshold distance where dodging is initiated, and a range of distances for which the angular speed is approximately zero, corresponding to the distance range in which linear fleeing is exhibited.

Finally, we note that deciding whether a given behavior pattern can reasonably be referred to as adaptively unpredictable requires an operational definition of protean behavior, preferably one that can be expressed quantitatively. As yet, we have been unable to find an agreed-upon rigorous quantitative definition of protean behavior in the literature, which makes detecting proteanism in our simulations all the more difficult.

4.2 Applications in Entertainment

In the short-to-middle-term future, we believe that the market with the biggest potential for commercial application of our techniques is leisure and entertainment software (rather than, e.g., autonomous mobile robots). Co-evolutionary techniques are already being used to develop sensory-motor controllers and morphologies for virtual/software agents whose behavior makes them fun to watch or interact with, or makes them easy to choreograph in producing movie sequences involving groups of complex interacting agents (e.g. [13, 14]).

To indicate the possibilities, we have produced 3D movie sequences of the trajectories shown in Figures 2, 4, 6, and 8. The movies were made by taking the trajectory data shown earlier and using this to move two virtual 3D agents around a 2D plane: the animats are given 3D 'bodies' and 'eyes', purely for visual effect: Figure 11 shows one frame from one of the movies. A 'virtual camera operator' tracks the animats by moving the 'camera' in 3D as the chase unfolds, according to some simple rules. Of course, whether the movie sequences are entertaining is a matter of subjective opinion. But we think it is fair to say that they are a *much* more stimulating way of presenting the data than the 2D trajectory plots given here. Readers with access to the world-wide-web can find the movies (encoded as MPEG sequences) at the URL: `http://www.cogs.susx.ac.uk/users/davec/pe.html`.

4.3 Monitoring Techniques

Whether we are interested in co-evolution of pursuit and evasion for scientific purposes or for reasons of making fun software, there are still unresolved issues in monitoring and characterizing co-evolutionary dynamics. The indications are that experiments in co-evolution of artificial autonomous agents are, in general, likely to be highly resource-intensive (i.e take a lot of time and/or a lot of computing power: a run of 1000 generations typi-

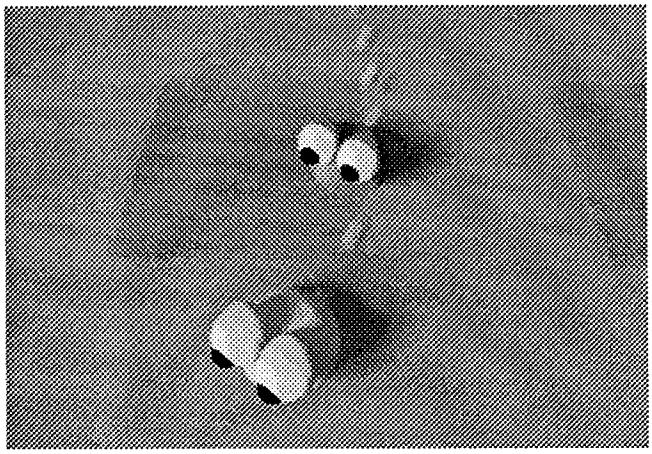


Figure 11: A frame from a computer-graphics pursuit-evasion 'entertainment' movie sequence: generation 999, same trajectory as Figure 6, 4.1sec into the trial.

cally takes about 3 weeks on an unladen Sun Sparc 20). As increasing magnitudes of resources are committed to such experiments, so there will be an increasing requirement for robust and informative monitoring techniques. The primary pragmatic need will be for techniques which make it possible to determine whether the evolutionary process is heading in the desired direction, or any direction at all. We discussed this problem and some tentative solutions in [6]. Our work in this area continues.

5 Conclusions

This paper has described our methods and presented tiny cross-sections of data from the torrent of megabytes that a complex co-evolutionary simulation can generate. But most of what we've learned from this project can't be expressed in numbers. We started out with several beliefs that turned out to be naive, simplistic, or simply mistaken; these included the conceits that (1) smart pursuers and evaders with ingenious nervous systems will evolve quickly under coevolution; (2) protean (adaptively unpredictable) behavior will emerge quickly, robustly, and permanently in almost any pursuit-evasion simulation; (3) almost any robot physics will lead to interesting pursuit-evasion strategies; (4) co-evolutionary progress is easy to measure; and (5) the details of evaluation functions, trial initialization methods, network parameters, morphogenesis methods, and genetic algorithm parameters won't matter much because coevolution is so robust. We know better now, and hope that other researchers will remain inspired by the grace and drama of real animals doing pursuit and evasion, but that they will be as cautious as we have become about the ability of computer simulation to capture the evolutionary dynamics that produced such behavior.

Acknowledgements

DC's work was supported in part by a grant from the UK EPSRC. GFM's work was supported by NSF-NATO Post-Doctoral Fellowship RCD-9255323, NSF Research Grant INT-9203229, the University of Sussex, the Univerity of Nottingham, and the Max Planck Society.

References

[1] R. D. Beer. On the dynamics of small continuous-time recurrent neural networks. *Adaptive Behavior*, 3(4):471–511, 1995.

[2] V. Braitenberg. *Vehicles: Experiments in Synthetic Psychology*. MIT Press Bradford Books, Cambridge MA, 1984.

[3] D. Cliff. NCAGE: Network control architecture genetic encoding. Technical Report CSRP325, University of Sussex School of Cognitive and Computing Sciences, 1994.

[4] D. Cliff, I. Harvey, and P. Husbands. Explorations in evolutionary robotics. *Adaptive Behavior*, 2(1):71–108, 1993.

[5] D. Cliff, I. Harvey, and P. Husbands. Artificial evolution of visual control systems for robots. In M. Srinivisan and S. Venkatesh, editors, *From Living Eyes to Seeing Machines*. Oxford University Press, 1996. In Press.

[6] D. Cliff and G. F. Miller. Tracking the Red Queen: Measurements of adaptive progress in co-evolutionary simulations. In F. Morán, A. Moreno, J. J. Merelo, and P.Chacón, editors, *Advances in Artificial Life: Proc. Third Euro. Conf. Artificial Life*, pp.200–218. Springer-Verlag, 1995.

[7] J. J. Grefenstette. The evolution of strategies for multiagent behaviors. *Adaptive Behavior*, 1(1):65–89, 1992.

[8] M. F. Land and T. S. Collett. Chasing behaviour of houseflies (*Fannia canicularis*). *Journal of Comparative Physiology*, 89:331–357, 1974.

[9] G. F. Miller. Protean primates: The evolution of adaptive unpredictability in competition and courtship. In A. Whiten and R. W. Byrne, editors, *Machiavellian Intelligence II*. Oxford University Press, 1996. In Press.

[10] G. F. Miller and D. Cliff. Protean behavior in dynamic games: Arguments for the co-evolution of pursuit-evasion tactics. In D. Cliff, P. Husbands, J.-A. Meyer, and S. Wilson, editors, *From Animals to Animats 3: Proc. Third Int. Conf. Sim. Adapt. Behav. (SAB94)*, pp.411–420. MIT Press, 1994.

[11] G. F. Miller and J. J. Freyd. Dynamic mental representations of animate motion: The interplay among evolutionary, cognitive, and behavioral dynamics. Technical Report CSRP 290, University of Sussex School of Cognitive and Computing Sciences, 1993.

[12] M. A. Potter and K. A. De Jong. A cooperative coevolutionary approach to function optimization. In Y. Davidor and H.-P. Schwefel, editors, *Proc. Third Conference on Parallel Problem Solving from Nature*, pp.249–257. Springer-Verlag, 1994.

[13] C. Reynolds. Competition, coevolution, and the game of tag. In R. Brooks and P. Maes, editors, *Artificial Life IV*, pp. 59–69. MIT Press 1994.

[14] K. Sims. Evolving 3D morphology and behavior by competition. In R. Brooks and P. Maes, editors, *Artificial Life IV*, pp. 28–39. MIT Press, 1994.

[15] D. Stork, B. Jackson, and S. Walker. 'Non-Optimality' via pre-adaptation in simple neural systems. In C. Langton, C. Taylor, J. D. Farmer, and S. Rasmussen, editors, *Artificial Life II*, pp.409–429. Addison Wesley, 1992.

[16] G. M. Werner and M. G. Dyer. Evolution of herding behaviours in artificial animals. In J.-A. Meyer, H. Roitblat, and S. Wilson, editors, *Proc. Second Int. Conf. Sim. Adapt. Behav. (SAB92)*, pp.393–399. MIT Press, 1993.

INCREMENTAL SELF-IMPROVEMENT FOR LIFE-TIME MULTI-AGENT REINFORCEMENT LEARNING

Jieyu Zhao **Jürgen Schmidhuber**

IDSIA, Corso Elvezia 36, CH-6900-Lugano, Switzerland
jieyu,juergen@idsia.ch - http://www.idsia.ch

Abstract

Previous approaches to multi-agent reinforcement learning are either very limited or heuristic by nature. The main reason is: each agent's or "animat's" environment continually changes because the other learning animats keep changing. Traditional reinforcement learning algorithms cannot properly deal with this. Their convergence theorems require repeatable trials and strong (typically Markovian) assumptions about the environment. In this paper, however, we use a novel, general, sound method for multiple, reinforcement learning "animats", each living a single life with limited computational resources in an unrestricted, changing environment. The method is called "incremental self-improvement" (IS — Schmidhuber, 1994). IS properly takes into account that whatever some animat learns at some point may affect learning conditions for other animats or for itself at any later point. The learning algorithm of an IS-based animat is embedded in its own policy — the animat cannot only improve its performance, but in principle also improve the way it improves etc. At certain times in the animat's life, IS uses reinforcement/time ratios to estimate from a single training example (namely the entire life so far) which previously learned things are still useful, and selectively keeps them but gets rid of those that start appearing harmful. IS is based on an efficient, stack-based backtracking procedure which is guaranteed to make each animat's learning history a history of long-term reinforcement accelerations. Experiments demonstrate IS' effectiveness. In one experiment, IS learns a sequence of more and more complex function approximation problems. In another, a multi-agent system consisting of three co-evolving, IS-based animats chasing each other learns interesting, stochastic predator and prey strategies.

1 Introduction

The problem. Some reinforcement learning algorithms are guaranteed to converge in certain limited, non-changing environments, e.g., (Kumar and Varaiya, 1986; Barto, 1989; Watkins and Dayan, 1992; Williams, 1992; Jaakkola et al., 1995; Kaelbling et al., 1995; Ring, 1994). All of them require the expected reinforcement (reward) for a given behavior to remain constant. In one way or another, all of them *assume that it is possible to collect a large amount of statistics from many exactly repeatable trials.* Consider, however, a typical co-evolutionary scenario where various learning "animats" compete with each other for (usually delayed) reinforcement. As they learn, each animat's environment and each animat's expected reinforcement for a given behavior tend to change due to changes in the other animats' behaviors. This means that, for any given animat, perhaps there is no such thing as an exactly repeatable trial. Previous algorithms don't provide principled (non-heuristic) approaches to such situations.

The questions. Constantly changing environments require us to rethink a bit the way we measure performance. First we observe that only if some animat's behavior conveys in some sense useful information about useful future behavior then it may hope to profit from previous experience. The questions are: at a given time in its single life, how can the animat estimate from a single training example (namely its entire life so far) which previously learned things will remain useful? How can it selectively keep them but get rid of those that start appearing harmful under the changed conditions?

Central idea (Schmidhuber, 1994) — details in section 2.2. To address these questions in a principled way, we will introduce the concept of a reinforcement/time ratio. At a given time in the life of each animat — it has got only one life, see, e.g., (Russell and Wefald, 1991; Boddy and Dean, 1994; Berry and Fristedt, 1985; Gittins, 1989; Greiner, 1996) for related views — there is *only one single training example* to evaluate the current long-term usefulness of any given previous modification of its behavior, namely the average reinforcement per

time since that modification occurred (the modification typically will have been computed by the animat's learning algorithm). At certain times in system life called "checkpoints", such singular observations are used by a stack-based backtracking method which invalidates certain previous behavioral modifications, such that the history of still valid modifications corresponds to a history of long-term reinforcement accelerations. Until the next checkpoint, the straight-forward generalization assumption is: each modification that until now appeared to have contributed to an overall speed-up will keep appearing useful.

For instance, whenever the environment appears to change in the sense that the reward per time for the current behavior is observed to decrease, the backtracking method will selectively undo those previously learned behavior modifications that do not appear useful any more (perhaps because they were too specifically tailored to previous situations) but will selectively keep those modifications that still appear useful in the sense that they until now were followed by long-term reinforcement accelerations (despite possible changes of the environment). Details in section 2.2.

Incremental self-improvement (details in section 2.1). An intriguing side-effect is: if we allow each animat to modify itself, by embedding the way it modifies its behavior within the behavior itself (this is easy to implement, see section 2.1), then the scheme will keep only those self-modifications followed by reinforcement speed-ups, in particular those leading to "better" future self-modifications, etc., recursively: embedding the learning mechanism within the modifiable system behavior does not involve any circularity, and the approach remains theoretically sound. The animats used in the experiments actually are of the "self-modifying" kind.

2 Implementation Details

There are many possible ways of implementing the basic principles above. The next paragraph provides an overview of our particular implementation.

Overview. To improve/speed up its own (initially very dumb, highly random) learning strategy, each animat makes use of an assembler-like programming language suitable to modify its own *stochastic policy*. The stochastic policy actually is an array of conditional, modifiable probability distributions (initially maximum entropy distributions) on a set of assembler-like instructions. There is one such distribution for each possible "program address". These distributions serve to compute the probability of the assembler-like instruction (and its actual parameters) to be executed next (this probability also depends on the current position of an "instruction pointer", which in turn depends on previously executed instructions). Many instructions (including instructions for adding, subtracting, multiplying, conditional jumps, etc.) may change the contents of so-called work cells, which are *never* reset: work cells are like an external notebook and are viewed as part of the policy environment. The animat may execute so-called "self-delimiting self-modification sequences" (SMSs). SMSs are instruction subsequences (generated according to the current policy) whose beginnings and ends are marked by special instructions. With the help of special "self-referential" instructions, SMSs can individually address and modify each variable probability distribution, thus being able to compute almost arbitrary policy modifications (which actually are sequences of probability modifications). However, to prevent animat life from becoming deterministic, SMSs are not allowed to assign *zero* probability to any instruction. Policy modifications can be computed only by SMSs. SMSs affect the probabilities of future SMSs, which affect the probabilities of future SMSs, etc. These recursive effects are taken care of by a stack-based backtracking method: whenever an SMS computes a policy modification, we push onto a stack all information required to compute at later times *the ratio between cumulative reinforcement and time since the modification was generated*, and to restore the original probability distributions if necessary. The backtracking method is invoked at certain times called "checkpoints". It *guarantees* that the animat regularly satisfies the so-called reinforcement acceleration criterion (RAC): at the end of each call of the backtracking procedure, the *still valid* policy modifications will correspond to an (in the worst case empty) history of long-term reinforcement accelerations (measured up until the current time in system life). In principle, the system can learn to deal with arbitrary reward delays by influencing the durations of its SMS and the checkpoint positions.

Outline of remainder of section 2. Subsection 2.1 will describe how each animat's policy is represented as a set of variable probability distributions on a set of assembler-like instructions, how the policy builds the basis for generating and executing a lifelong instruction sequence, and how the system can modify itself executing special "self-referential instructions". Subsection 2.2 then will formally specify the animat's goal (namely, to maximize cumulative reinforcement to be obtained during its single life), and its reinforcement acceleration method (performance evaluation and backtracking using a stack).

2.1 Policy and Program Execution of a Single Animat

Storage / Instructions. Each animat makes use of an assembler-like programming language similar to the one in (Schmidhuber, 1995b). It has n addressable *work cells* with addresses ranging from 0 to $n-1$. The variable, real-valued contents of the work cell with address

k are denoted c_k. Processes in the external environment occasionally write inputs into certain work cells. There also are m addressable *program cells* with addresses ranging from 0 to $m-1$. The variable, integer-valued contents of the program cell with address i are denoted d_i. An internal variable *Instruction Pointer* (*IP*) with range $\{0,\ldots,m-1\}$ always points to one of the program cells (initially to the first one). There also is a fixed set I of n_{ops} integer values $\{0,\ldots,n_{ops}-1\}$, which sometimes represent instructions, and sometimes represent arguments, depending on the position of *IP*. For each value j in I, there is an assembler-like instruction B_j with N_j integer-valued parameters. In the following (incomplete) list of instructions to be used in experiment 1, the symbols $a1, a2, a3$ stand for parameters that may take on integer values between 0 and $n-1$ (later we will encounter additional instructions):

B_0: *Add(a1, a2, a3)* : $c_{a3} \Leftarrow c_{a1} + c_{a2}$ (add the contents of work cell *a1* and work cell *a2*, write the result into work cell *a3*).

B_1: *Sub(a1, a2, a3)* : $c_{a3} \Leftarrow c_{a1} - c_{a2}$.

B_2: *Mul(a1, a2, a3)* : $c_{a3} \Leftarrow c_{a1} * c_{a2}$.

B_3: *Mov(a1, a2)* : $c_{a2} \Leftarrow c_{a1}$.

B_4: *Restart: IP*$\Leftarrow 0$ *(jump back to 1st program cell).*

Later (in the experimental sections) we will encounter additional primitives allowing the animat (1) to move around in an environment, and (2) to perceive positions of obstacles and other animats.

Instruction probabilities / Current policy. For each program cell i there is a variable probability distribution P_i on I. For every possible $j \in I$, $(0 \leq j \leq n_{ops}-1)$, P_{ij} specifies for cell i the conditional probability that, when pointed to by *IP*, its contents will be set to j. The set of all current P_{ij}-values defines a probability matrix P with columns P_i $(0 \leq i \leq m-1)$. P is called the animat's *current policy*. In the beginning of animat life, all P_{ij} are equal (maximum entropy initialization). If $IP = i$, the contents of i, namely d_i, will be interpreted as instruction B_{d_i} (such as *Add* or *Mul*), and the contents of cells that immediately follow i will be interpreted as B_{d_i}'s arguments, to be selected according to the corresponding P-values. For example, the integer sequence 1 6 8 7 will be interpreted as *Sub(6, 8, 7)* — subtract the contents of cell 6 from the contents of cell 8 and put the result into cell 7.

"Self-reference". To obtain an animat that can explicitly modify its own policy (by running its own learning strategies), we introduce a special "self-referential" instruction *IncProb* not yet mentioned above:

B_5: *IncProb(a1, a2, a3)* : Increase P_{ij} by γ percent, where $i = c_{a1} * n_{ops} + c_{a2}$ and $j = c_{a3}$ (this construction allows for addressing a broad range of program cells), and renormalize P_i (but prevent P-values from falling below a minimal value ϵ, to avoid total determinism). In the experiments, we will use $\gamma = 15, \epsilon = 0.001$.

In conjunction with other primitives, *IncProb* may be used in instruction sequences that compute directed policy modifications. Calls of *IncProb* represent the *only* way of modifying the policy.

Self-delimiting self-modification sequences (SMSs). Another yet unmentioned "self-referential" primitive is the parameterless instruction

B_6: *EndSelfMod()*. Temporarily disable *IncProb*, by preventing future *IncProb* instructions from causing any probability modifications, until N_r additional nonzero reinforcement signals have been received.

The first *IncProb* after animat "birth" or after each "checkpoint" (see next subsection) begins an SMS. The SMS ends itself by executing *EndSelfMod()*. While an SMS is running, the animat's policy is protected from performance evaluations (to be treated in section 2.2). Some of the (initially highly random) action subsequences executed during animat life will indeed be SMSs. Depending on the nature of the other instructions, SMSs can compute almost arbitrary sequences of modifications of P_{ij} values, resulting in almost arbitrary modifications of context-dependent probabilities of future action subsequences, including future SMSs. *Policy changes can be generated only by SMSs.* Policy changes will also *influence* the probabilities of future SMSs.

"Meta-learning" potential. SMSs build the basis for "meta-learning": SMSs are generated according to the policy, and may change the policy. Hence, the policy can essentially change itself, and also the way it changes itself, etc. There is no explicit difference between "learning" and "meta-learning" and "meta-meta-learning".

Instruction cycle. The animat forever executes the following basic loop:

1. Randomly generate an integer $j \in I$ according to probability distribution (matrix column) P_{IP} (the distribution of the program cell pointed to by *IP*). Set program cell contents $d_{IP} := j$. Translate j into the corresponding current instruction B_j. Look up the number N_j of cells required to store B_j's parameters. If $IP > m - N_j - 1$, reset *IP* to 0, go to step 1. Otherwise generate instruction arguments for the N_j cells immediately following *IP* according to their probability distributions P_{IP+1}, ..., P_{IP+N_j}, and set *IP* to $IP + N_j + 1$.

2. If the current instruction fails to pass a syntax check (e.g., inappropriate parameters), go to 1.

3. If there is a "checkpoint" (details in the next subsection), e.g., if the current instruction is the first *IncProb* of an SMS, then possibly undo certain previous probability modifications (using the backtracking method to be introduced in the next subsection) to ensure that the current history of still valid policy modifications is a success story.

4. Execute the current instruction. This may change the environment, or work cell contents, or the instruction pointer (e.g., in case of jumps), or even the policy P itself (in case the instruction is an *IncProb*).

5. If the current instruction is *IncProb*, then push information about the changes it caused onto a stack (the information in the stack will be used at later times to decide whether it is necessary to countermand certain policy modifications — see details in the next subsection).

6. Goto 1.

2.2 *How a Single Animat Accelerates Reinforcement.*

The previous subsection just described a particular animat whose policy is able to modify itself in a certain sense. We have not yet talked about the precise goal of the animat, and how it learns to keep "good" self-modifications and discard others. This is discussed in the current subsection, which paves the way for understanding what needs to be done during steps 3 and 5 of the animat's infinite basic loop above.

The animat's goal. Occasionally, the environment provides real-valued "reinforcement". The sum of *all* reinforcements obtained between "animat birth" (at time 0) and time $t > 0$ is denoted by $R(t)$. Throughout its life-time, the animat's goal is to maximize $R(T)$, the cumulative reinforcement at (unknown) "death" T.

Reinforcement/time ratios based on single observations. Since the environment may change in an unknown way (no guarantee of exactly repeatable trials), at a given time t in animat life, we have only one single "training example" to evaluate the current long-term usefulness of any given previous SMS, namely the *average reinforcement per time since that SMS occurred.* Let us denote the a-th SMS in animat life by SMS_a, which starts execution at time t_a^1, generates a policy modification (a sequence of probability modifications) denoted M_a, and completes itself (by executing *EndSelfMod*) at time $t_a^2 > t_a^1$. For $t \geq t_a^2$ and $t \leq T$, the reinforcement/time ratio $Q(a,t)$ is defined as

$$Q(a,t) = \frac{R(t) - R(t_a^1)}{t - t_a^1}.$$

Obviously, the computation of reinforcement/time ratios takes into account all computation time. The success of SMS_a recursively depends on the success of later SMS_b, $b > a$: the cumulative reinforcement collected after SMS_a includes the cumulative reinforcement collected during and after SMS_b, $b > a$. In particular, performance improvements include those improvements that make future additional improvements more likely: policy modification M_a can prove its long term usefulness by setting the stage for additional, useful modifications $M_b, b > a$, etc. Recall that there is no explicit difference between "learning" and "meta-learning" and "meta-meta-learning".

Reinforcement delays. As we will see below, the first evaluations of SMS_a's performance will be delayed at least until after its end. While SMS_a is still running, it is in a "grace period" which may be used to collect *delayed* reinforcement to justify M_a — this is important when reinforcement signals occur long after policy modifications took place. We will see that since SMS beginnings and ends are defined by special instructions, the animat is in principle able to *learn* how long to wait for delayed reinforcement.

Currently valid modifications. After t_a^2, SMS_a's modifications of the policy will stay in existence until (a) being overwritten by later SMS_b, $b > a$, or until (b) being *invalidated* by the stack-based method to be described below. To define *valid modifications*, only (b) is relevant: after t_a^2, the policy modification M_a generated by SMS_a will remain *valid* as long as the stack-based method below does not undo M_a (by restoring the previous policy right before SMS_a started).

Reinforcement acceleration criterion (RAC). Our animat's goal will be to regularly achieve RAC, which is satisfied if the beginning of each completed SMS that computed a still valid modification has been followed by long-term reinforcement acceleration — measured up until the current time. More formally: RAC is satisfied at time t, if for each SMS_u that computed a *still valid* modification M_u (if there is any) of the animat's policy,

$Q(u,t) > \frac{R(t)}{t}$ and for all $v < u$, where SMS_v computed a still valid M_v: $Q(u,t) > Q(v,t)$.

The method to achieve RAC. An initially empty stack is used to store information about currently valid policy changes computed by SMSs. Occasionally, at times called *"checkpoints"*, this information is used to restore previous policies, such that RAC holds. The method is based on two complementary kinds of processes: pushing and checking (Schmidhuber, 1994, 1995a, 1996a,b).

(1) PUSHING. Suppose the animat uses *IncProb* to modify one of its probability distributions in step 3 of the basic loop from section 2.1. Suppose that the current SMS is SMS_a, which started at time t_a^1 (and will end at t_a^2). If the current *IncProb* is the first *IncProb*

of SMS_a, then in step 5 (left unspecified in section 2.1) we push t_a^1 and $R(t_a^1)$ (which will be needed to compute $Q(a,t)$ values at later times t) onto the stack. In any case, we push the address of the modified program cell, its old probability distribution right before the current modification (represented by n_{ops} real values), and a pointer to the first stack entry of SMS_a (to keep track of SMS_a's beginning — recall that M_a actually is a *sequence* of probability modifications). After t_a^2, the modification M_a will remain *valid* as long as all this information will remain on the stack.

(2) CHECKING (the procedure below) occurs at each **"checkpoint"**. Checkpoints occur regularly during animat life, but never outside step 3 of the basic loop from section 2.1. A checkpoint definitely occurs right before some SMS_a starts by executing its first *IncProb* instruction, right before time t_a^1, such that, by definition, t_a^1 coincides with the end of the checking process:

> **WHILE** no time t is reached such that one of conditions (1-3) holds
>
> > (1) $Q(l,t) > Q(l',t)$, where $l > l'$, and M_l and $M_{l'}$ are the two most recent currently valid modifications (if existing),
> >
> > (2) $Q(l,t) > \frac{R(t)}{t}$, where M_l is the only currently valid modification (if existing),
> >
> > (3) the stack is empty.
>
> **DO:** pop the information about the most recent valid modification M_c off the stack, and use the popped probability distributions to *invalidate* M_c, by restoring the corresponding previous policy.

The time consumed by pushing processes, checking processes, and all other computations is taken into account (for instance, time goes on as popping takes place).

Theoretical soundness. Using induction, it can be shown that this backtracking procedure ensures that RAC holds after each checking process (Schmidhuber, 1994, 1995a, 1996a,b). At each "checkpoint", the animat will get rid of M_a if t_a^1 was not followed by long-term reinforcement speed-up (note that before countermanding M_a, the animat will already have countermanded all $M_b, b > a$). No modification M_a is guaranteed to remain valid forever. The animat will keep only those self-modifications followed by long-term reinforcement speed-ups, in particular those leading to "better" future self-modifications, etc., recursively ("incremental self-improvement"). Note also that a "self-modifying" policy can in principle *learn* to influence checkpoint positions.

Example. A simple example of life-time reinforcement acceleration is shown in Figure 1. All checkpoints are marked by "x". The current time t is a checkpoint, too. Suppose the animat has just finished its checking process. Modifications M_1, M_2 and M_4 (computed by self-modification sequences SMS_1, SMS_2 and SMS_4, respectively) are currently valid, because they satisfy RAC as follows: $Q(4,t) > Q(2,t) > Q(1,t) > \frac{R(t)}{t}$.

Modification M_3 (computed by SMS_3), however, was invalidated at the previous checkpoint t'. The reason was: M_3 prevented RAC from being satisfied, because $Q(3,t') \leq Q(2,t')$. M_2, however, remained valid, because $Q(2,t') > Q(1,t')$ at t'.

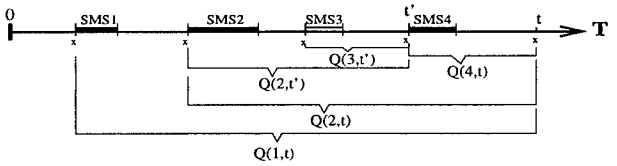


Figure 1: Satisfying the reinforcement acceleration criterion.

Typically, a change of environment that makes the current policy less successful will be reflected by decreasing reinforcement time ratios, and the animat will selectively undo those previously learned policy modifications that do not appear useful any more (perhaps because they were too specifically tailored to previous tasks).

Generalization assumption. After a checking process, *until the next checkpoint*, the straight-forward generalization assumption is: modifications that survived the most recent checking process (because they appeared to contribute to speed-up of average reinforcement intake) will remain useful. In general, unknown, changing environments, which other generalization assumption would make sense? Recall that since life is one-way (time is never reset), at each checkpoint the animat has to generalize from a *single* experience concerning the usefulness of any given previous learning process: the average reinforcement per time since that learning process occurred. Learning from singular experiences contrasts other, less realistic kinds of reinforcement learning, which, in one way or another, require the assumption that it is possible to collect voluminous statistics from *repeatable* trials.

3 Experiment 1: Function Approximation

Our first experiment does not yet involve multiple animats. It just demonstrates that IS can successfully learn in a changing environment where the tasks to be solved get more and more difficult over time. The experiment is intended to serve as an introduction to the next experiment (section 4), where there will be multiple, coevolving animats with conflicting goals.

Goal. Our system is exposed to a sequence of more and more complex function approximation problems.

The functions to be learned are $f_1(x,y) = x + y$; $f_2(x,y,z) = x + y - z$; $f_3(x,y,z) = (x + y - z)^2$; $f_4(x,y,z) = (x + y - z)^4$; $f_5(x,y,z) = (x + y - z)^8$.

Trials. The system's single life can be separated into n successive trials A_1, A_2, ..., A_n. The i-th trial lasts from discrete time step $t_i + 1$ until discrete time step t_{i+1}, where $t_1 = 0$ (system birth) and $t_{n+1} = T$ (system death). In a given trial A_i we first select a function $g_i \in \{f_1, \ldots, f_5\}$. As the trial number increases, so does the probability of selecting a more complex function. In early trials the focus is on f_1. In late trials the focus is on f_5. In between there is a gradual shift in task difficulty.

Once g_i is selected, randomly generated real values x, y and z are then put into work cells 0, 1, 2, respectively. The trial is finished as soon as the system performs a write operation on work cell 6, whose contents are to be interpreted as the system's response. If c_6 fulfills the condition $|g_i(x,y,z) - c_6| < 0.0001$, then the current reward becomes 1.0; otherwise the current reward is 0.0.

Instructions. Instruction sequences can be composed from the following primitive instructions (compare section 2.1): *Add(a1,a2,a3), Sub(a1,a2,a3), Mul(a1,a2,a3), Mov(a1,a2), IncProb(a1,a2,a3), EndSelfMod(), Restart()*. Each instruction occupies 4 successive program cells (some of them unused if the instruction has less than 3 parameters). We use $m = 49, n = 7$.

Checkpoints. Checkpoints occur after each 5th consecutive non-zero reinforcement signal after the end of each SMS, i.e., we set $N_r = 5$.

Huge search space. Given the primitives above, random search would require about 10^{17} trials on average to find a solution for f_5 — the search space is huge. The gradual shift in task complexity, however, helps IS to learn f_5 much faster, as will be seen below.

Results. After about 9.4×10^8 instruction cycles (ca. 10^8 trials), the system was able to compute f_5 almost perfectly, given arbitrary real-valued inputs. The corresponding speed-up factor over (infeasible) random or exhaustive search is about 10^9 — compare paragraph "Huge search space" above. The solution (see Figure 2) involved 21 strongly modified probability distributions of the policy (after learning, the correct instructions had extreme probability values). At the end, the most probable code was given by the following integer sequence:
1 2 1 6 1 0 6 6 2 6 6 6 2 6 6 6 2 6 6 6 4 * * *...

The corresponding "program" and the (very high) probabilities of its instructions and parameters are shown in Table 1.

During its life the system generated a lot of self-modifications to compute the strongly modified policy. This includes changes of the probabilities of self-modifications! It is quite interesting (and also quite difficult) to find out to which extent the system uses self-modifying instructions to learn how to use self-modifying instructions. Figure 3 gives a vague idea of what's going on by showing a typical plot of the frequency of *IncProb* instructions during system life (sampled at intervals of 10^6 instruction cycles). Soon after its birth, the system found it useful to dramatically increase the frequency of *IncProb*; near its death (when there was nothing more to learn) it significantly reduced this frequency.

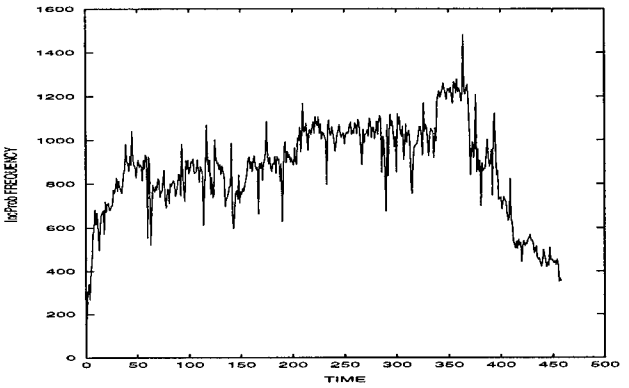


Figure 3: *Numbers of executed self-modifying instructions plotted against time, sampled at intervals of* 10^6 *instruction cycles. The graph reflects that the system soon uses self-modifying instructions to increase the frequency of self-modifying instructions. Near system death the system learns that there is not much to learn any more, and decreases this frequency.*

Stack evolution. The temporary ups and downs of the stack reflect that as the tasks change, the system selectively keeps still useful old modifications (corresponding to information conveyed by previous tasks that is still valuable for solving the current task), but deletes modifications that are too specific for previous tasks. In the end, there were only about 200 stack entries corresponding to only 200 *valid* probability modifications – this is a small number compared to the about $5 * 10^5$ self-modifications executed during system life.

4 Experiment 2: Predator and Prey

Scenario. In our second experiment there are three co-evolving, IS-based animats A, B, C. Each animat simultaneously is both predator and prey. A's prey is B. B's prey is C. C's prey is A. A's predator is C. B's predator is A. C's predator is B. Each animat's goal is to catch its prey as often (and as quickly) as possible, while evading its predator. Each time it catches its prey, it receives a reinforcement of 1.0; each time it gets caught, it receives negative reinforcement -1.0 and is randomly reset to a new position. Each animat's environment changes because the other animats in its environment learn and change. We have a symmetric zero-sum game, resulting in a continuous evolutionary race.

The simulated environment consists of an area of 600×

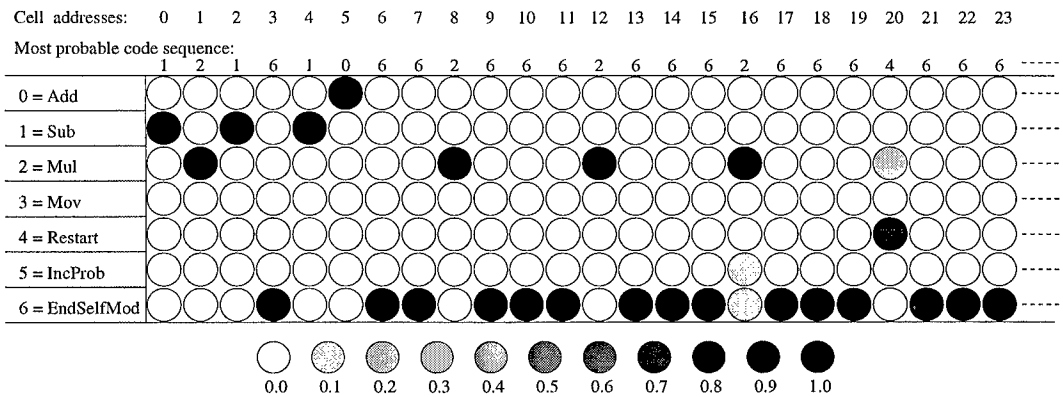


Figure 2: *The final state of the probability matrix for the function learning problem. Grey scales indicate the magnitude of probabilities of instructions and parameters. The matrix was computed by self-modification sequences generated according to the matrix itself (initially, all probability distributions were maximum entropy distributions).*

	Probabilities	Instruction	Parameters	Semantics
1.	(0.994, 0.975, 0.991, 0.994)	Sub	(2, 1, 6)	$(z-y) \Longrightarrow c_6$
2.	(0.994, 0.981, 0.994, 0.994)	Sub	(0, 6, 6)	$(x-(z-y)) \Longrightarrow c_6$
3.	(0.994, 0.994, 0.994, 0.994)	Mul	(6, 6, 6)	$(x+y-z)^2 \Longrightarrow c_6$
4.	(0.994, 0.994, 0.994, 0.994)	Mul	(6, 6, 6)	$(x+y-z)^4 \Longrightarrow c_6$
5.	(0.869, 0.976, 0.994, 0.994)	Mul	(6, 6, 6)	$(x+y-z)^8 \Longrightarrow c_6$
6.	(0.848, —, —, —)	Restart	(–, –, –,)	$0 \Longrightarrow IP$

Table 1: The final, most probable "program" and the corresponding probabilities.

800 pixels. See Figure 4. Obstacles are set in advance but can be changed by the experimenter at any later time. Each animat has circular shape and a diameter of 30 pixel widths. At a given time, it can move up to 9 steps in 8 different directions. Directions are represented as integers in $\{0,\ldots,7\}$: 0 for north, 1 for northeast, 2 for east,... etc.. A single step places the animat's center at the closest pixel 5 pixel widths away in the current direction. Each animat is equipped with limited "active" sight: by executing certain actions, it can sense obstacles or its prey or its predator within up to 10 steps in front of it. It also can make turns relative to prey and predator position. Since each animat can reach any position in the field (except those blocked by obstacles), the multi-agent system's entire state space is *huge*. Each animat's instruction set includes the following $n_{ops} = 10$ instructions:

B_0: *Move(n)* — move n $(0 \leq n \leq 9)$ steps forward in the current direction.

B_1: *Turn(d)* — change current direction D to $(D + d + 4) mod 8$, where $(0 \leq d \leq 7)$.

B_2: *TurnRelativeToPredator(d)* — First turn to the direction that best matches the line connecting the centers of animat and predator, then *Turn(d)* $(0 \leq d \leq 7)$.

B_3: *TurnRelativeToPrey(d)* — (analogous to B_2).

B_4: *LookForPredator(n)* — if predator is not within $n+1$ steps in front of the animat $(0 \leq n \leq 9)$, then increase IP by 4 (this is a limited kind of conditional jump).

B_5: *LookForPrey(n)* — (analogous to B_4).

B_6: *LookForObstacle(n)* — (analogous to B_4).

B_7: *Restart* — jump back to program cell 0 (see section 2.1.)

B_8: *EndSelfMod* — End the current SMS (see instruction B_6 in section 2.1).

B_9: *IncProb(a1, a2, a3)* : See instruction B_5 in section 2.1.

Each instruction occupies 2 successive program cells (the second one unused if there are no parameters) except for *IncProb*, which occupies 4. We set $m = 50, n = 10$.

Given the primitives above, each animat faces a complex partially observable Markov decision problem (POMDP), e.g., (Schmidhuber, 1991; Whitehead, 1992; Jaakkola et al., 1995; Kaelbling et al., 1995; Ring, 1994; McCallum, 1993; Littman et al., 1995; Wiering and Schmidhuber, 1996) — the current input by itself does not necessarily provide all information needed to determine the optimal next action.

Checkpoints. Like in section 3., for each animat, a checkpoint occurs after each 5th consecutive non-zero reinforcement signal following each SMS (i.e $N_r = 5$).

Copying successful policies. To achieve a balanced evolutionary race with more or less equal opponents, we let animats with inferior performance occasionally "steal" the best animat's policy. The corresponding copying procedure is initiated at special times called "copypoints". Once a copypoint occurs, we wait until all animats have encountered their respective next checkpoints and have finished the corresponding checking processes; until then, no animat may start another SMS by itself (*IncProb* disabled). Then we make each animat's policy equal to the policy of the currently most successful animat among the three. The proper way of doing this is to act as if the corresponding sequence of modifications caused to the policies of the two less successful animats was generated by standard SMSs — the old probability distributions are pushed onto the respective stacks, together with information about the current time and the reinforcement so far. Copypoints occur every 10^6 instruction cycles.

Results. After each animat executed 10^9 instruction cycles, all three animats (whose behavior can be observed directly on the screen) exhibit quite interesting pursuit-evasion behaviors. They all chase each other, using existing obstacles trying to avoid being caught. The final number of valid probability modifications per animat is reflected by the final stack size, which is about 350. Figure 5 shows a snapshot with recent movement traces, but unfortunately the often surprising and complex dynamics cannot be conveyed by a single snapshot. Since the game is open-ended (unlike the one from section 3), no animat reduces the frequency of its self-modifications (as in Figure 3) but keeps learning and reacting to changes of the others' behavior. Each animat's strategy appears more complex than the one of the single system from section 3, but also harder to analyze. In particular, although each animat quickly replaces the initial high-entropy probability distributions of its policy by low-entropy distributions, it often leaves significant probability values for alternative behavioral sequences. Much of each animat's policy is not committed to any obvious deterministic strategy, but is stochastic instead.

However, there are policy fragments making sense to a human observer. For instance, Figure 5 depicts a policy fragment that makes the following action sequence likely to occur:

1. *If there is no obstacle within 6 steps ahead, then skip instructions 2 and 3.* **2.** *Execute a random instruction (whose parameter is likely to be 5).* **3.** *Face the prey (the parameter determining relative direction to the prey is set to zero).* **4.** *Move 8 steps forward.* **5.** *If the prey cannot be seen within 9 steps, then skip instructions 6 and 7.* **6.** *If (at the next time step) the prey cannot be seen within 6 steps ahead, then skip instructions 7 and 8 (in conjunction with instruction 5, instruction 6 may be used to test whether the prey is getting closer from the vision periphery).* **7.** *Move 8 steps forward (this instruction will be executed, e.g., if the prey seems close or getting closer).* **8.** *If the predator cannot be seen within 9 steps ahead, then skip instructions 9 and 10.* **9.** *If the predator cannot be seen within 8 steps ahead, then skip instructions 10 and 11 (in conjunction with instruction 8, instruction 9 may be used to test whether the predator is getting closer from the vision periphery).* **10.** *Face the prey (this instruction will typically turn the animat away from the predator — but note that the behavior of the animat's prey and its predator also depend on each other in a complicated manner).* **11.** *Jump back to program cell 0.* **12.** *... a lot of additional, partly highly deterministic code.*

Although this fragment does not exactly look like an exercise in elegant programming, it represents part of a strategy that works reasonably well for a broad range of situations.

5 Conclusion

Traditional reinforcement learning algorithms cannot properly deal with situations where each learning animat's environment continually changes, e.g., because of other learning, changing animats. Animats based on incremental self-improvement (IS), however, have a principled way of dealing with changing environments. The learning strategies of IS-based animats are embedded in their own policies and thus accessible to self-modifications. IS-based animats occasionally compute reinforcement/time ratios to estimate from a single training example (namely the entire life so far) whether certain previous policy modifications are still useful (perhaps some aren't any longer because of the changing environment). To compute these ratios, the entire previous life-time is (at least implicitly) considered by a backtracking method which regularly, selectively undoes those previously learned policy modifications that do not appear useful any more (perhaps because they were too specifically tailored to previous tasks). Experiments confirm that IS-based animats are indeed able to successfully learn in continually changing environments.

The principles of IS are very general. They are in no way limited to assembler-like programming languages with very simple primitive instructions such as the ones used in this paper. Primitive actions can actually be almost anything. For instance, an action may correspond to a Bayesian analysis of previous events. While this analysis is running, *time is running*, too. Thus, the complexity of the Bayesian approach is automatically taken into account. Or, as in (Wiering and Schmidhuber, 1996), actions may be calls of Levin search (Levin, 1973), a theoretically optimal algorithm for a wide variety of *non-incremental* search problems. Or, actions may be calls of a Q-learning variant. This makes sense in situations where the applicability of Q-learning is questionable because the environment does not satisfy the preconditions that would make Q-learning sound.

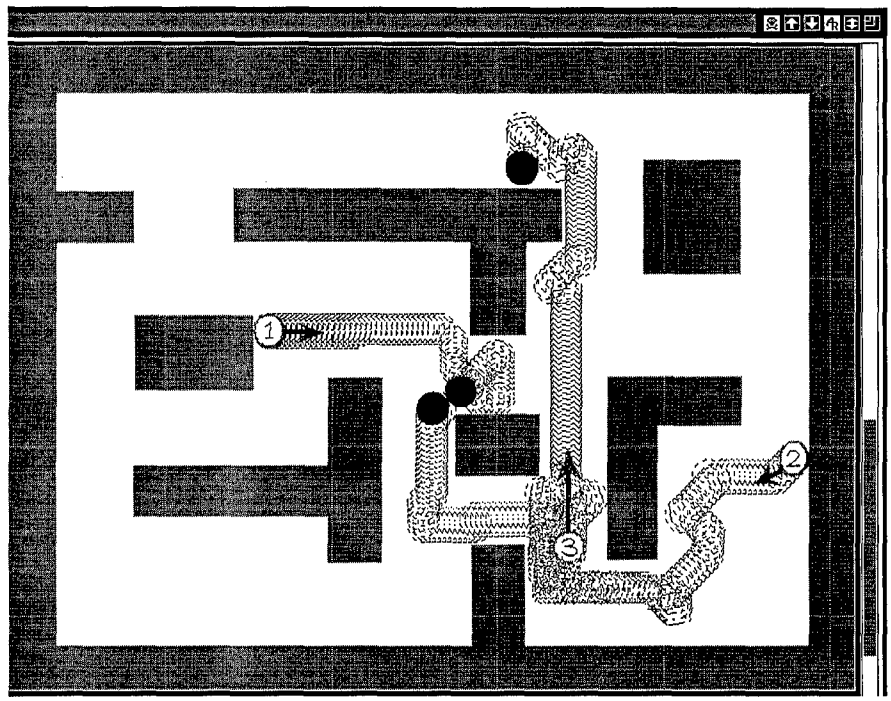


Figure 4: *Snapshot taken during the lifelong pursuit-evasion game. Animats A, B, C come from initial positions 1, 2, 3, respectively. Arrows indicate initial directions of each animat. Animat C successfully escapes its predator B by quickly moving north, partly because B turns to avoid contact with its predator A, which was on its way east towards B but then turned away when its predator C passed by. This turn provides additional benefits for A, because of B's own turn: while heading in the general direction of C, B gets caught by A.*

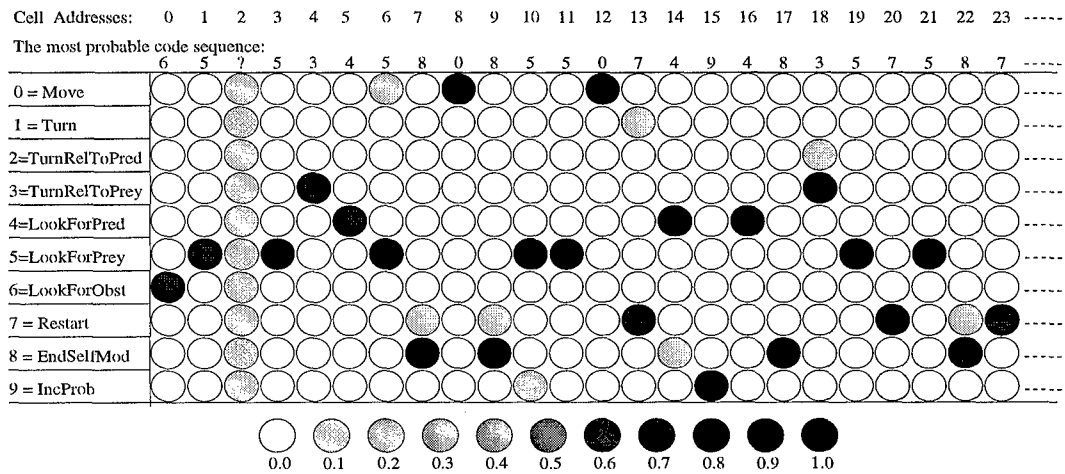


Figure 5: *Columns 0-23 of the final probability matrix for the predator/prey game. It was computed by self-modification sequences generated according to the probability matrix itself (initially, all probability distributions were maximum entropy distributions). We may say that, at a given time in system life, the probability matrix embeds its own current stochastic learning strategy, and also its own "meta-learning" strategy, etc..*

Acknowledgments

This work was supported by SNF grant 21-43'417.95 "Incremental Self-Improvement".

References

Barto, A. G. (1989). Connectionist approaches for control. Technical Report COINS 89-89, University of Massachusetts, Amherst MA 01003.

Berry, D. A. and Fristedt, B. (1985). *Bandit Problems: Sequential Allocation of Experiments.* Chapman and Hall, London.

Boddy, M. and Dean, T. L. (1994). Deliberation scheduling for problem solving in time-constrained environments. *Artificial Intelligence*, 67:245–285.

Gittins, J. C. (1989). *Multi-armed Bandit Allocation Indices.* Wiley-Interscience series in systems and optimization. Wiley, Chichester, NY.

Greiner, R. (1996). PALO: A probabilistic hill-climbing algorithm. *Artificial Intelligence*, 83(2).

Jaakkola, T., Singh, S. P., and Jordan, M. I. (1995). Reinforcement learning algorithm for partially observable Markov decision problems. In Tesauro, G., Touretzky, D. S., and Leen, T. K., editors, *Advances in Neural Information Processing Systems 7, to appear.* MIT Press, Cambridge MA.

Kaelbling, L., Littman, M., and Cassandra, A. (1995). Planning and acting in partially observable stochastic domains. Technical report, Brown University, Providence RI.

Kumar, P. R. and Varaiya, P. (1986). *Stochastic Systems: Estimation, Identification, and Adaptive Control.* Prentice Hall.

Levin, L. A. (1973). Universal sequential search problems. *Problems of Information Transmission*, 9(3):265–266.

Littman, M., Cassandra, A., and Kaelbling, L. (1995). Learning policies for partially observable environments. Technical report, Brown University, Providence RI.

McCallum, R. A. (1993). Overcoming incomplete perception with utile distinction memory. In *Machine Learning: Proceedings of the Tenth International Conference.* Morgan Kaufmann, Amherst, MA.

Ring, M. B. (1994). *Continual Learning in Reinforcement Environments.* PhD thesis, University of Texas at Austin, Austin, Texas 78712.

Russell, S. and Wefald, E. (1991). Principles of Metareasoning. *Artificial Intelligence*, 49:361–395.

Schmidhuber, J. (1995a). Environment-independent reinforcement acceleration. Technical Note IDSIA-59-95, IDSIA. Invited talk at Hongkong University of Science and Technology.

Schmidhuber, J. (1996a). A general method for incremental self-improvement and multi-agent learning in unrestricted environments. In Yao, X., editor, *Evolutionary Computation: Theory and Applications.* Scientific Publ. Co., Singapore.

Schmidhuber, J. (1996b). A general method for multi-agent learning in unrestricted environments. In *Adaptation, Co-evolution and Learning in Multiagent Systems, Technical Report SS-96-01*, pages 84–87. American Association for Artificial Intelligence, Menlo Park, Calif.

Schmidhuber, J. H. (1991). Reinforcement learning in Markovian and non-Markovian environments. In Lippman, D. S., Moody, J. E., and Touretzky, D. S., editors, *Advances in Neural Information Processing Systems 3*, pages 500–506. San Mateo, CA: Morgan Kaufmann.

Schmidhuber, J. H. (1994). On learning how to learn learning strategies. Technical Report FKI-198-94, Fakultät für Informatik, Technische Universität München. Revised January 1995.

Schmidhuber, J. H. (1995b). Discovering solutions with low Kolmogorov complexity and high generalization capability. In Prieditis, A. and Russell, S., editors, *Machine Learning: Proceedings of the Twelfth International Conference*, pages 488–496. Morgan Kaufmann Publishers, San Francisco, CA.

Watkins, C. J. C. H. and Dayan, P. (1992). Q-learning. *Machine Learning*, 8:279–292.

Whitehead, S. (1992). *Reinforcement Learning for the adaptive control of perception and action.* PhD thesis, University of Rochester.

Wiering, M. and Schmidhuber, J. (1996). Solving POMDPs with Levin search and EIRA. In Saitta, L., editor, *Machine Learning: Proceedings of the Thirteenth International Conference.* Morgan Kaufmann Publishers, San Francisco, CA. To appear.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256.

Dynamics of Co-evolutionary Learning

Hugues Juillé **Jordan B. Pollack**
Computer Science Department
Volen Center for Complex Systems
Brandeis University
Waltham, MA 02254-9110
{hugues, pollack}@cs.brandeis.edu

Abstract

Co-evolutionary learning, which involves the embedding of adaptive learning agents in a fitness environment which dynamically responds to their progress, is a potential solution for many technological chicken and egg problems, and is at the heart of several recent and surprising successes, such as Sim's artificial robot and Tesauro's backgammon player. We recently solved the two spirals problem, a difficult neural network benchmark classification problem, using the genetic programming primitives set up by [Koza, 1992]. Instead of using absolute fitness, we use a relative fitness [Angeline & Pollack, 1993] based on a competition for coverage of the data set. As the population reproduces, the fitness function driving the selection changes, and subproblem niches are opened, rather than crowded out. The solutions found by our method have a symbiotic structure which suggests that by holding niches open, crossover is better able to discover modular building blocks.

1 Introduction

Co-evolution is an ecological theory which attempts to explain how traits can evolve which are dependent between different species. In evolutionary computation however, it has been appropriated from its ecological roots to describe any iterated adaptation involving "arms-races", either between learning species or between a learner and its learning environment. Examples of co-evolutionary learning include the pioneering work by Hillis on sorting networks [Hillis, 1992], by Tesauro on self-playing Backgammon learner [Tesauro, 1992] with a recent follow up by Pollack, Blair and Land [Pollack et al., 1996], by Sims and Ray in evolving life-forms [Sims, 1994, Ray, 1992], by Angeline and Pollack on co-evolving Tic-tac-toe players [Angeline & Pollack, 1993]. In the adaptive behavior community, there is a focus developing on co-evolution in predator/prey games [Reynolds, 1994, Miller & Cliff, 1994].

Using competitive fitness in a massively parallel implementation of the genetic programming (GP) paradigm [Koza, 1992] we solved the problem of intertwined spirals, a very difficult classification benchmark from the field of neural networks. This learning problem, originated by Alexis Wieland, perhaps based on the cover of Perceptrons, has been a challenge for pattern classification algorithms and has been subject of much work in the AI community, in particular in the Neural Network field (e.g., [Lang & Witbrock, 1988, Fahlman & Lebiere, 1990, Carpenter et al., 1992]). In Neural Network classification systems, based on linear, quasi-linear, radial, or clustering basis function, the intertwined spirals problem leads to difficulty. When it is solved, the neural net solution often has a very "expansive" description of the spiral, i.e. the conjunction of many small regions, does not generalize outside the training regions, and is thus not particularly satisfying.

In this paper we compare our competitive fitness co-evolutionary approach to an absolute fitness approach to the spirals problem and find it more effective. Moreover, co-evolution in this context leads to interesting functional modularizations of the problem.

Section 2 presents a survey of the implementation of our Massively Parallel Genetic Programming (MPGP). This will help to understand the techniques that have been used in the following sections. Then, the intertwined spiral problem is described in section 3, along with its representation in the competitive fitness framework and results. Section 4 presents both theoretical and empirical analysis of our results comparing canonical and co-evolutionary optimization.

2 Massively Parallel GP

2.1 Parallel Evaluation of S-expressions

MPGP runs on a SIMD machine of 4096 processor elements (PEs), the MasPar MP-2. The individual structures that undergo adaptation in GP are represented by

expression trees composed from a set of primitive functions and a set of terminals (either variables or functions of no argument). Usually, the number of functions is small, and the size of the expression trees are restricted, in order to restrict the size of the search space.

In our parallel implementation, each of the 4096 processors simulates a virtual processor. This virtual processor is a *Stack Machine* and takes the postfix representation of an S-expression as its input.

To be able to evaluate a GP expression, the following instructions are supported by the abstract machine:

- one instruction for each primitive function of the function set. At execution time, arguments for these instructions are popped from the stack into general purpose registers, the function is computed, and the result is pushed on the top of the stack.
- a `PUSH` instruction which pushes on the top of the stack the value of a terminal,
- a `IFGOTO` and a `GOTO` instruction which are necessary for branching if conditional functions are used,
- a `STOP` instruction which indicates the end of the program.

This architecture allows each PE to process efficiently a different genetic program in a MIMD-like way. The parallel interpreter of the SIMD machine reads the current postfix instruction for each virtual processor and sequentially multiplexes each instruction, *i.e*, all processors for which the current instruction is a `PUSH` become active and the instruction is performed; other processors are inactive (*idle* state). Then, the same operation is performed for each of the other instructions in the instruction set in turn. Once a `STOP` instruction is executed for a processor, that processor becomes idle, leaving the result of its evaluation on the top of the stack. When all processors have reached their `STOP` instruction, the parallel evaluation of the entire population is complete.

[Perkis, 1994] has already shown that the stack-based approach for Genetic Programming can be very efficient.

2.2 *Models for Fitness Evaluation, Selection and Recombination*

The MasPar MP-2 is a 2-dimensional wrap-around mesh architecture. In our implementation, the population has been modeled according to this architecture: an individual or a sub-population is assigned to each node of the mesh and, therefore, has 4 neighbors. This architecture allows us to implement different models for fitness evaluation, selection and recombination, using the kernel of the parallel GP described in the previous section.

In this paper, only a tournament style of competitive evolution has been used and compared to canonical GP. A more general presentation of the different strategies that have been implemented can be found in [Juillé & Pollack, 1996].

3 The Spiral Problem and the Competitive Evolution Paradigm

3.1 *Presentation*

The intertwined spiral problem consists of learning to classify points on the plane into two classes according to two intertwined spirals. The data set is composed of two sets of 97 points, on the plane between -7 and +7. These two intertwined spirals are shown as "×" and "o" in figures 5 and 6.

[Koza, 1992] and [Angeline, 1995] have also investigated this problem using the Genetic Programming paradigm. We used the same setup as them to define the problem and to perform our experiments. That is, the function set is composed of: $\{+, -, *, \%, iflte, sin, cos\}$, and the terminal set is composed of: $\{x, y, \Re\}$, where $\Re$ is the ephemeral random constant.

With a population of 4096 individuals, we tried two different approaches to tackle this problem. In the first experiment, following Koza and Angeline, the fitness function was defined as the number of hits out of 194.

In the second experiment, the fitness was defined as the result of a competition among the individuals. We ignored the fact that we really knew the absolute fitness function, and set up a "game" in which only relative fitness was used as the basis for reproduction. The trivial idea would be to simply compare the absolute score of each individual and the winner would be the individual with the larger score. However, such a competition of absolute scores would simply approximate the canonical version.

Instead, we only counted a player's ability to classify those test cases which are *not* classified by its opponent. As more or less copies of a player spread through the population, their scores may rise or fall depending on how many other members of the population also "cover" the test cases. This is a form of adaptive behavior implemented dynamically in the fitness function. As a simplified view, consider a full pairwise evaluation between one weak but unique player with 25 novel hits, against 4 identical strong players all with the same 50 hits. Although they would reproduce twice as fast in an absolute fitness competition, in this modified tournament, they will only receive their 50 points for playing the weak player, who will actually receive 100! In section 4, a simplified ecological model is presented to study the dynamics of the population evolution when an absolute fitness or a competitive fitness is used to control interactions between species. We do not play all-against-all, but several rounds of a more limited tournament competition, and compute the final relative fitness of each individual as the sum of all its scores during the competition. We can

of course track the absolute fitness of a population even though it is not used otherwise.

Our hypothesis is that the competitive evolution would work better because it would promote more diversity in the population, and allow subpopulations which covered different subproblems to emerge. As copies of individuals which perform well on parts of the spiral spread through the population, they will start to meet themselves in competition, and get a score of 0. This allows other individuals who may have less total hits, but cover other parts of the spiral to survive. From the recombinations between individuals of those two sub-populations one may expect the emergence of a better individual that combine the "advantages" of both.

Several approaches may be used when simulating a competitive evolution [Sims, 1994]. In this work, each generation is composed of a sequence of competition rounds in which individuals are "randomly" paired up. In fact, because of the architecture of our parallel computer which doesn't have any fast-access shared-memory, this random pairing is approximated by making all the individuals perform a "random walk" in the population. At each round, the score of individuals is the number of hits that their opponent doesn't get. At the end of each generation, individuals' fitness is calculated by summing all their scores in the competition.

Once individual fitness is evaluated, selection and recombination are performed according to a fitness proportionate rule. Details of the implementation of this model of tournament, and of selection and recombination procedures for MPGP can be found in [Juillé & Pollack, 1996].

3.2 *Preliminary Results and Discussion*

For the two classes of experiments, we performed 25 runs and each run was stopped after 300 generations. At each generation, 90% of the population was replaced by offsprings resulting from recombination and the remaining 10% was the result of fitness proportionate reproduction. Each individual meets 96 opponents at each generation (this number comes from the implementation of the tournament on the mesh architecture of the MasPar).

Our preliminary results concerning performance illustrate that competitive evolution outperforms the absolute fitness approach. In fact, the absolute fitness strategy works better at the beginning, with average fitness rising faster. However, this absolute fitness paradigm improves its current solution very slowly after its initial burst of optimization, and is ultimately outperformed by the competitive evolution.

There are multiple competing explanations for this performance gap. It may be that the absolute fitness is simply converging prematurely. It may be that the competitive fitness system benefits from more diversity. In sections below we analyze and try to understand these differences.

```
(sin (% (iflte (- (- (- (* _A _A)
                        (sin
                         (% (iflte  -0.52381
                                    _B
                                    (sin -0.33333)
                                    -0.33333)
                            -0.33333)))
                    (* _B _B))
                 (% _A (% -0.33333 _A)))
              -0.80952
              _B
              (sin
               (% (% _A
                     (- (cos
                         (sin
                          (* (cos
                              (sin -0.52381))
                             (% _B
                                (% _A
                                   (- (cos
                                       -0.33333)
                                      0.04762))))))
                        0.04762))
                  (sin (sin -0.33333)))))
       -0.33333))
```

Figure 1: A 52-atom S-expression scoring 194 for the intertwined spiral problem.

If $(4 * x^2 - y^2) < 0.0$ **then**
    return $(sin(-3.0 * y))$;
else
    return $\left(sin(\frac{0.3214*x}{0.04762-cos(sin(\frac{y}{x}*0.7874))})\right)$;
endif

Figure 2: Interpretation of the solution for the intertwined spiral problem.

However, only a few runs of competitive fitness have provided us with a perfect (194 hits) solution for the intertwined spiral problem within 300 generations. We harvested some of the perfect classification solutions; One of the shortest of these S-expressions has 52 atoms and is shown in figure 1.

Because of the relatively small size of this result we were able to analyze it and simplify it mathematically, by collapsing constant calculations, removing insignificant digits, algebraic simplification, and elimination of redundant "introns". This analysis resulted in the conditional function presented in figure 2.

Basically, this solution splits the geometric plane into two domains and a different function is used for each domain. Figure 3 displays the $4x^2 - y^2$ function which multiplexes the two other functions, shown in figure 4, to create the spiral.

The resulting function is shown in figure 5, which plots

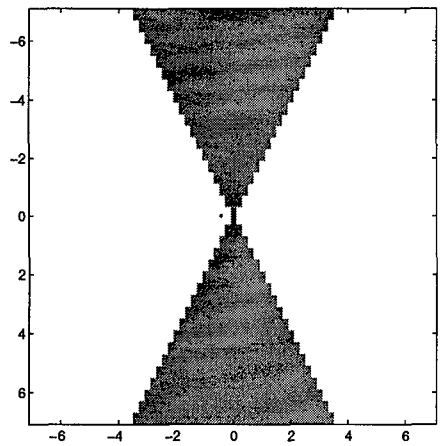


Figure 3: $4x^2 - y^2 < 0$, used to divide the plane into two domains.

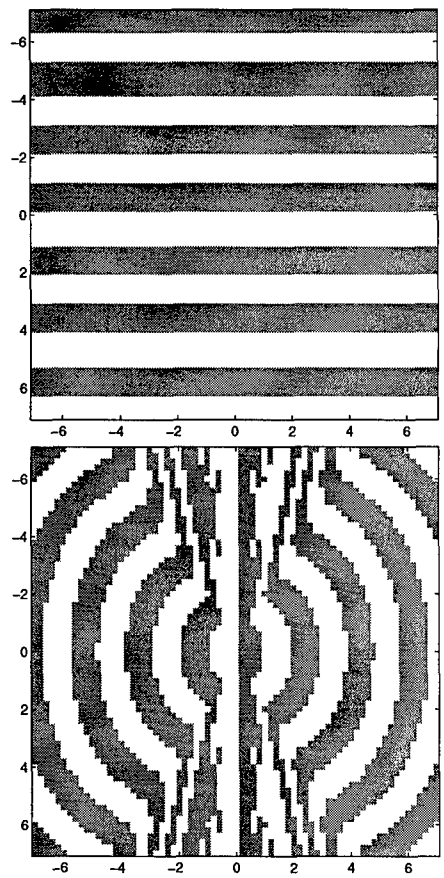


Figure 4: $sin(-3y)$ and the other function which are selectively added to make a spiral.

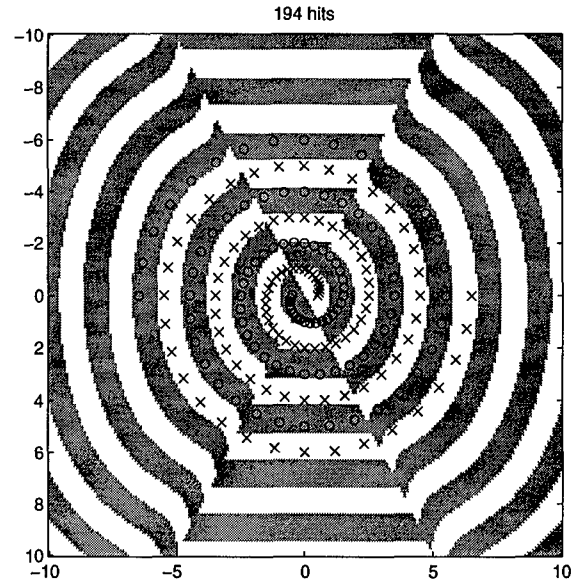


Figure 5: Perfect score generalizing classification of the two intertwined spirals.

the function (above/below 0) along with the training data on the range -10 to 10. Although it does not form a perfect spiral, it does continue to simulate a spiral way outside the original training range. In another set of experiments (limited to 100 generations) another perfect solution has been discovered (presented in figure 6). The S-expression representing this solution is composed of 161 atoms.

Furthermore, we believe that compared to neural network solutions, which are often the composition of hundreds of clusters or decision boundaries, and some of the GP solutions shown by Koza, ours is the most perspicacious to date. The fact that the spiral is composed of a symbiosis of two (or more) functions which cover separate parts of the data supports the hypothesis that the relative fitness competitive evolution strategy can be more effective than an absolute fitness function. This idea is supported by the analysis presented in the following section.

4 Co-evolution and the Dynamics of Learning

4.1 A Theoretical model for Absolute and Relative Fitness

To support the idea that competitive evolution allows subpopulations which cover different part of the problem to survive, contrary to an absolute fitness driven evolution, we propose the following analysis. The two-intertwined problem is a classification problem. Therefore, it can be seen as a set of test cases and the popu-

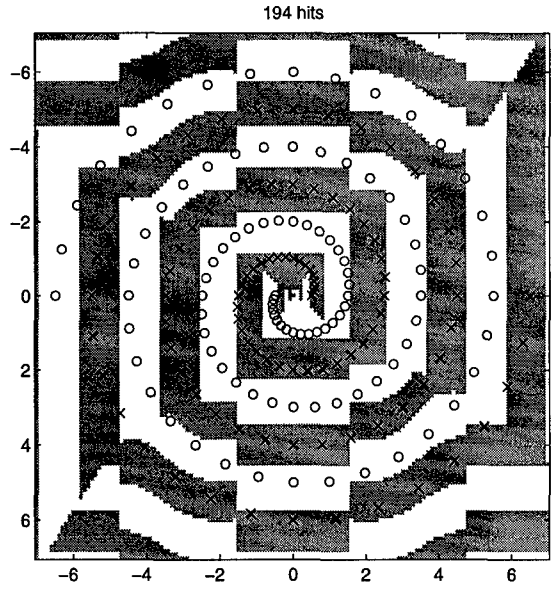


Figure 6: Another perfect score classification of the two intertwined spirals.

lation can be split up into groups (or clusters) in which individuals would cover exactly the same test cases. For the sake of clarity, let us formalize this idea. First, let us define the following terms:

- n: number of test cases,
- m: number of groups (or clusters) that compose the population,
- t_i: i^{th} test case,
- G_j: j^{th} group of individuals,
- $s_j(t)$: size (number of individuals) of group G_j at time t,
- $T(G_j)$: returns a list of booleans of size n in which the k^{th} entry indicates whether the test case t_k is covered by individuals in group G_j,
- $\mathcal{B}$: matrix whose rows are the $T(G_j)$s.

$$\mathcal{B} = \begin{pmatrix} T(G_1) \\ \vdots \\ T(G_m) \end{pmatrix}$$

Each entry $b_{i,j}$ of $\mathcal{B}$ is a 1 (*true*) if the test case t_j is covered by the group G_i, and 0 (*false*) otherwise.

For the following, let us consider an example:

- $n = 10$,
- $m = 5$,
- $T(G_1) = (0,1,1,1,0,1,0,1,0,1)$,
  $T(G_2) = (1,0,0,0,1,0,0,0,1,0)$,
  $T(G_3) = (0,1,0,1,0,0,1,1,0,1)$,
  $T(G_4) = (0,0,1,1,0,0,0,0,1,0)$,
  $T(G_5) = (0,1,0,1,1,0,1,0,0,1)$

Now, we can define the $(m \times m)$ square matrix $\mathcal{A}$ for which each entry $a_{i,j}$ equals the number of test cases correctly classified by group G_i but that group G_j doesn't. Thus, each entry of $\mathcal{A}$ is defined as follows:

$$a_{i,j} = \sum_{l=1}^{n} (b_{i,l} \wedge \neg b_{j,l})$$

With our example, $\mathcal{A}$ equals:

$$\mathcal{A} = \begin{pmatrix} 0 & 6 & 2 & 4 & 3 \\ 3 & 0 & 3 & 2 & 2 \\ 1 & 5 & 0 & 4 & 1 \\ 1 & 2 & 2 & 0 & 2 \\ 2 & 4 & 1 & 4 & 0 \end{pmatrix}$$

Now, we can define the fitness function for the two cases of study:

- *absolute fitness* for an individual of group G_j:

$$f_a(j) = \sum_{l=1}^{n} b_{jl}$$

For our example:
$f_a(1) = 6; f_a(2) = 3; f_a(3) = 5; f_a(4) = 3; f_a(5) = 5$

- *relative fitness* for an individual of group G_j:

$$f_r(j) = \sum_{l=1}^{m} (s_l(t) \times a_{j,l})$$

According to this definition, each individual competes once against all other individuals in the population. In our experiments, we only approximate this by making each individual compete against a sample of the population.

For the sake of simplicity, we assume there are no recombination between individuals but only fitness proportionate reproduction. Indeed, what we want to show with this simplified model is that subpopulations that cover different test cases survive when competitive evolution is involved. Therefore, we want to study the dynamics of the evolution of group size with time. A simple rule for fitness proportionate reproduction, similar to the one used by [Lindgren, 1992] to model population evolution, gives us:

$$s_j(t+1) = s_j(t) \times \left(1 + \alpha \times \frac{f(j) - \overline{f}}{\overline{f}}\right)$$

where:

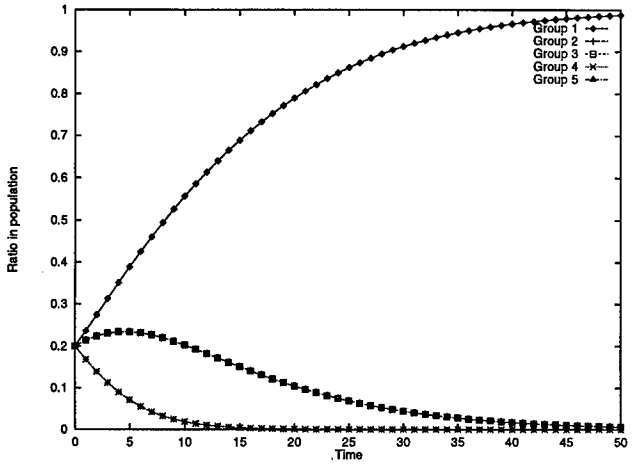


Figure 7: Evolution of the ratio for each group in the population in the case of an absolute fitness.

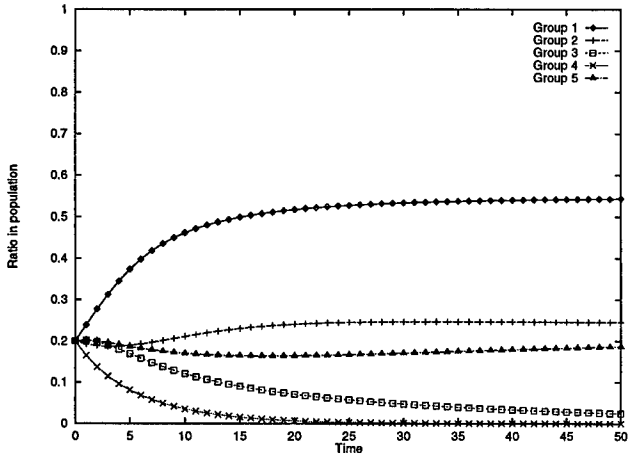


Figure 8: Evolution of the ratio for each group in the population in the case of a relative fitness.

- α is a parameter that controls the speed of the simulated evolution,
- $f(j)$ is the fitness. According to the case of study, it is replaced by $f_a(j)$ or $f_r(j)$.
- $\overline{f}$ is the average of the fitness.

A normalization step for $s_j(t+1)$ is then performed in order to keep a constant population size. If $\alpha = 1$, we get the well known expression for fitness proportionate reproduction:

$$s_j(t+1) = s_j(t) \times \frac{f(j)}{\overline{f}}$$

The graphical results of the evolution of the ratio for each group in the population for the two models of evolution are presented in figure 7 and figure 8. For our analysis, all groups have the same size at $t = 0$, and we took $\alpha = 0.5$. One can see that in the case of the absolute fitness, all the population is overcome by the first group which has the largest absolute fitness ($f_a(1) = 6$). The curves for the groups 2 and 4, and for the groups 3 and 5 overlap in this figure. On the contrary, in the case of the competitive evolution, once stability is reached, the first group takes a little more than 50% of the population and groups 2 and 5 around 20%. Group 4 disappears very quickly and group 3 takes only a tiny part of the population. It is possible to prove that these ratios at the equilibrium are independent of the initial size of the different groups (at the condition that no group has null size) and of the value of the non-null parameter α.

The aim of this analysis is to show that competitive evolution allows different subpopulation to survive, contrary to the canonical model of evolution, therefore keeping more diversity in the population. We also believe that in the case of the intertwined spiral problem, recombination of individuals from different subpopulations are at the origin of new solutions that cover some part of the problem that were specific to each of the two subpopulations. This idea is confirmed by the results of experiments presented in section 4.2.

4.2 Diversity and Useful Recombination

We realized that our co-evolution system was in fact operating to prevent convergence by increasing diversity in the population. This can also be done simply by changing parameters in a canonical genetic optimization task. So we performed more experiments to compare the evolutionary and the co-evolutionary approaches. In particular, we tried three different settings for the parameters that control the convergence rate of the search procedure and therefore control the decrease of diversity in the population. More precisely, the *normalized fitness* of individuals, which is used to control the selection process, is computed as follows:

$$raw_fitness = score$$

$$standardized_fitness = (max(score) - raw_fitness) \times \alpha$$

$$adjusted_fitness = \frac{1}{1+standardized_fitness}$$

$$normalized_fitness = \frac{adjusted_fitness}{\sum_{population} adjusted_fitness}$$

The parameter α controls the range of the standardized fitness and therefore the distribution of the normalized fitness. Indeed, if α decreases, the difference between fit and less fit individuals for the adjusted fitness, and therefore for the normalized fitness, decreases, making the convergence slower. However, the raw fitness doesn't represent the same measure for the absolute and relative fitness approaches (the number of hits for the former and the number of hits not covered by each of the opponents for the later). Thus, the only way to compare the two methods is to try a large range for the parameter α. For

Absolute fitness	Relative fitness
$\alpha = 1.0$	$\alpha = 0.2$
$\alpha = 0.2$	$\alpha = 0.05$
$\alpha = 1.0$ $(standardized_fitness)^2$	$\alpha = 0.01$

Table 1: Parameter setting for the experiments.

the absolute fitness, we tried two values for α and we performed one experiment for which the standardized fitness was squared, altering in another way the distribution of the normalized fitness. For the relative fitness experiments, three values were tested for α. Table 1 presents the different parameter settings for our experiments.

The tournament-like competition was implemented to be a realistic model of competition. In particular, it could be used to co-evolve game strategies [Angeline & Pollack, 1993]. However, in the case of an inductive learning problem like the intertwined spirals, the set of test cases is well-defined, fixed and of manageable size. Therefore, it is possible to efficiently implement an *all vs. all* competition as follows:

- For each test case, compute the number $\overline{S_i}$ of individuals that do *not* classify it correctly. This can be implemented in $O(\log n)$ on a parallel machine, using a form of divide-and-conquer to perform addition (*reduce* operator). Then, $\overline{S_i}$ is made available to all the individuals.

- The relative fitness of an individual j is then:

$$\sum_{i=1}^{\#\ test\ cases} b_{j,i} \times \overline{S_i}$$

  where: $b_{j,i}$ equals 1 if individual j classifies correctly the i^{th} test case, and 0 otherwise.

The result of this process is the same as if each individual would have compete against all the other individuals in the population.

We addressed the issue of all vs. all competition here for the following reason. For most problems, the tournament competition is of more practical interest than all vs. all competition. This is the case in particular when the set of test cases is too large to allow an exhaustive evaluation of individuals with the whole set (*e.g.*, when each individual represents a game strategy). Thus, it is interesting to estimate how accurately the tournament competition approximates the all vs. all competition.

We limited the number of generations to 150. The results are presented in figure 9 where each curve corresponds to the average over 25 runs with the same value for the parameters.

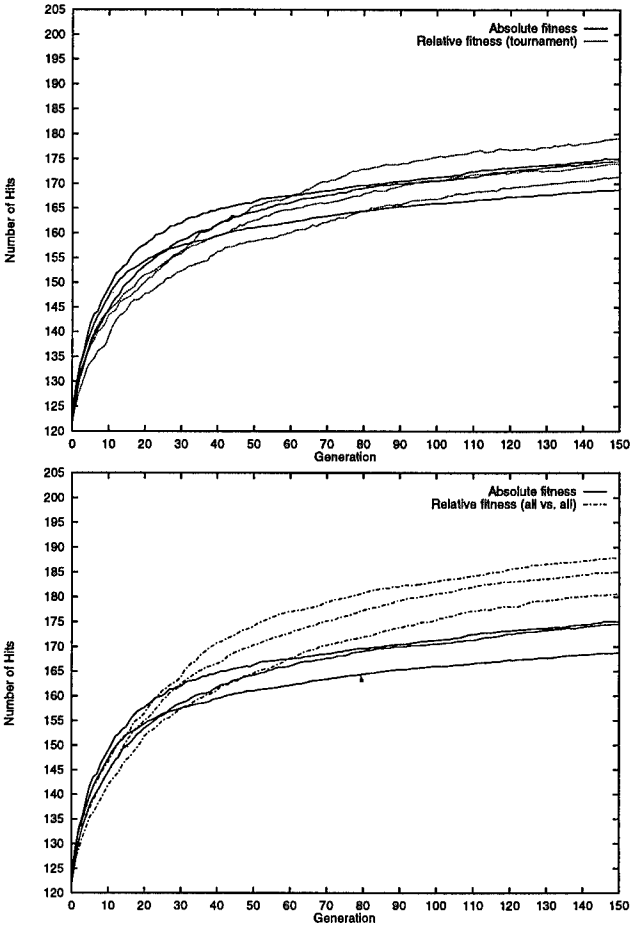


Figure 9: Absolute fitness versus tournament-like competition (top) and absolute fitness against all vs. all competition (bottom) for the intertwined spiral problem, for different parameter setting. Each curve is the average of the best individual, at each generation, over 25 runs.

The first observation is that all vs. all competition clearly outperforms canonical evolution. After 150 generations, one of the three parameter settings resulted in a perfect solution for 5 out of the 25 runs. Those runs where extended up to 200 generations and resulted in 10 perfect solutions out of the 25 runs. None of our experiments with canonical evolution resulted in a perfect solution. In the case of tournament-like competition, it is more difficult to conclude, even if a slight advantage might be given to this form of co-evolution. Only 2 out of the 75 runs resulted in a perfect solution before 150 generations.

Those experiments show that co-evolution offers a different approach to tackle the learning task and potentially works better than canonical evolution. The tournament-like competition uses only 96 rounds (compared to 4096 rounds for all vs. all) which might be seen as too small. It is difficult to extrapolate the number of rounds to achieve a given accuracy for the approximation

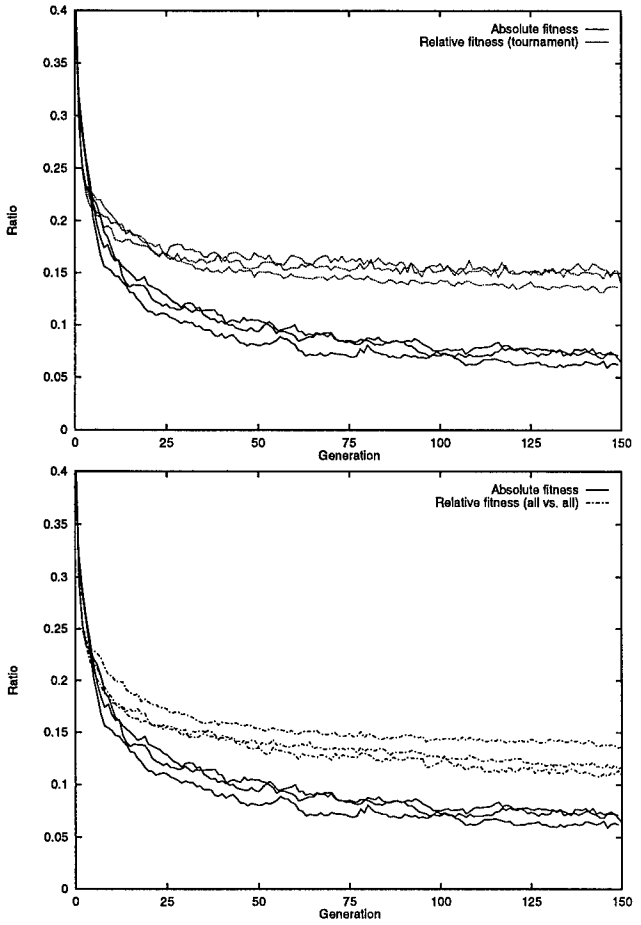


Figure 10: Comparison between absolute fitness and tournament-like competition (top) and between absolute fitness and all vs. all competition (bottom) for the evolution of the ratio of offsprings in the population that cover more test cases than their parents, for different parameter settings.

of all vs. all but it seems that 96 rounds is a good compromise regarding computer resource when the method presented in this paper to perform all vs. all cannot be applied.

For the same experiments, we also observed the evolution of another measure in order to show that learning is more efficient in the case of co-evolution. Rosca [Rosca & Ballard, 1996] defined *differential fitness* as a measure of the fitness improvement in the population. He defined this measure for offspring i as follows:

$$Differential_Fitness(i) = Standard_Fitness(i) - min_{p \in Parents(i)}\{Standard_Fitness(p)\}$$

For an heuristic reason [Rosca & Ballard, 1996], the *min* of the parents was taken to define the differential fitness. However, in our case, one is more interested in a measure that would indicate an improvement of offsprings over both parents. Therefore, the *max* of the parents has been taken to define our differential fitness. Moreover, in order to compare both approaches, we defined *Standard_Fitness* as the number of hits. This definition is used only for the evaluation of the differential fitness and is independent of the previously defined standard fitness used to evaluate the normalized fitness. Thus, the differential fitness is positive only for those offsprings that cover more test cases than both parents. Figure 10 presents the evolution of the ratio of offsprings in the population for which this new differential fitness is positive. Each curve represents the average over 25 runs. Data were collected while performing the previous experiments.

One can see that the average ratio seems to be almost independent of the parameter setting for relative as well as absolute fitness. However, this ratio for co-evolution is at least 50% larger than for canonical evolution after the first 50 generations. This difference means that the probability that an offspring be better than both its parents, *i.e.* the probability of *useful* recombination or the probability of the exchange of building blocks, is significantly greater in the case of a relative fitness than for an absolute fitness. Indeed, since co-evolution maintains niches that cover different subsets of the test cases and that the relative fitness favours the covering of all the test cases by those subsets, it is more likely that recombination will occur between parents that cover different test cases. This is not the case for the absolute fitness which doesn't have this kind of bias. Moreover, *diversity* in the population is not the main reason to explain this difference since the increase of diversity for experiments with absolute fitness doesn't change significantly the ratio. This clearly shows that our model of co-evolution favors useful recombination and, ultimately, is more likely to lead to a better solution than canonical evolution.

5 Conclusion

Experiments presented in this paper show that the classification procedure for a challenging problem (namely, the intertwined spiral problem) can be significantly improved by using a relative fitness rather than an absolute fitness approach. As we look at the All versus All tournament, which is not plausible for most tasks, we can see that the competitive fitness model of co-evolution approximates fitness sharing [Goldberg & Richardson, 1987], which helps prevent premature convergence by increasing diversity in a population. Rosin and Belew's [Rosin & Belew, 1995] work on fitness sharing demonstrates this connection as well.

In summary, we achieve co-evolution within a single species by using a heuristic which dynamically adapts the fitness function to help the population cover all the test cases, rather than giving a great reproductive advantage to those individuals which perform well on the whole

problem. Indeed, by giving the minority an advantage in the competition, one can expect the emergence of novel niches which are held open until they are incorporated into more successful solutions.

In evolutionary terms, the set of traits which lead to increased relative fitness of individuals or species cannot be pre-determined in a finite list as can be done for the intertwined spirals problem. However, the traits which arise over evolutionary time scales, such as strength, speed, olfaction, vision, and even cognition, are simply a competitive advantage until competitors acquire those traits or find counter-measures. Thus co-evolution, which may be viewed as an adaptive behavior on evolutionary time scales between species, can also be useful for problems for which absolute fitness is known on individual species, such as a population of genetic programs.

References

[Angeline, 1995] Angeline, P. J. (1995). Two self-adaptive crossover operations for genetic programming. In *Advances in Genetic Programming II.* MIT Press.

[Angeline & Pollack, 1993] Angeline, P. J. & Pollack, J. B. (1993). Competitive environments evolve better solutions for complex tasks. In *The Fifth International Conference on Genetic Algorithms*, pp. 264–270. Morgan Kaufmann.

[Carpenter et al., 1992] Carpenter, G., Grossberg, S., Markuzon, N., Reynolds, J., & Rosen, D. (1992). Fuzzy artmap: A neural network architecture for incremental supervised learning of analog multidimensional maps. *IEEE Transactions on Neural Networks*, 3:698–713.

[Fahlman & Lebiere, 1990] Fahlman, S. E. & Lebiere, C. (1990). The cascade-correlation learning architecture. In Touretzky (Ed.), *Advances in Neural Information Processing Systems 2.* Morgan Kauffman.

[Goldberg & Richardson, 1987] Goldberg, D. E. & Richardson, J. J. (1987). Genetic algorithms with sharing for multimodal function optimization. In Grefenstette, J. J. (Ed.), *Genetic Algorithms and their Applications: Proceedings of the Second International Conference on Genetic Algorithms*, pp. 41–49. Lawrence Erlbaum Associates.

[Hillis, 1992] Hillis, W. D. (1992). Co-evolving parasites improve simulated evolution as an optimization procedure. In Langton, C. et al. (Eds.), *Artificial Life II*, pp. 313–324. Addison Wesley.

[Juillé & Pollack, 1996] Juillé, H. & Pollack, J. B. (1996). Massively parallel genetic programming. In Angeline & Kinnear (Eds.), *Advances in Genetic Programming II.* MIT Press.

[Koza, 1992] Koza, J. R. (1992). *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* MIT Press.

[Lang & Witbrock, 1988] Lang, K. J. & Witbrock, M. J. (1988). Learning to tell two spirals apart. In *Proceedings of the 1988 Connectionist Summer Schools.* Morgan Kaufmann.

[Lindgren, 1992] Lindgren, K. (1992). Evolutionary phenomena in simple dynamics. In Langton, C. et al. (Eds.), *Artificial Life II*, pp. 295–312. Addison Wesley.

[Miller & Cliff, 1994] Miller, G. F. & Cliff, D. (1994). Protean behavior in dynamic games. In Cliff, D., Husbands, P., Meyer, J., & Wilson, S. (Eds.), *From Animals to Animats 3: Proceedings of the Third International Conference on Simulation of Adaptive Behavior.* MIT Press.

[Perkis, 1994] Perkis, T. (1994). Stack-based genetic programming. In *Proceedings of the 1994 IEEE World Congress on Computational Intelligence.* IEEE Press.

[Pollack et al., 1996] Pollack, J. B., Blair, A. D., & Land, M. (1996). Coevolution of a backgammon player. To appear in the proceedings of the Fifth Artificial Life Conference.

[Ray, 1992] Ray, T. S. (1992). An approach to the synthesis of life. In Langton, C. et al. (Eds.), *Artificial Life II*, pp. 371–408. Addison Wesley.

[Reynolds, 1994] Reynolds, C. W. (1994). Competition, coevolution, and the game of tag. In *Proceedings of the Fourth Artificial Life Conference.* MIT Press.

[Rosca & Ballard, 1996] Rosca, J. P. & Ballard, D. H. (1996). Discovery of subroutines in genetic programming. In Angeline & Kinnear (Eds.), *Advances in Genetic Programming II.* MIT Press.

[Rosin & Belew, 1995] Rosin, C. D. & Belew, R. K. (1995). Methods for competitive co-evolution: Finding opponents worth beating. In Eshelman, L. J. (Ed.), *Proceedings of the Sixth International Conference on Genetic Algorithms*, San Mateo, California. Morgan Kauffmann.

[Sims, 1994] Sims, K. (1994). Evolving 3d morphology and behavior by competition. In Brooks & Maes (Eds.), *Artificial Life IV*, pp. 28–39. MIT Press.

[Tesauro, 1992] Tesauro, G. (1992). Practical issues in temporal difference learning. *Machine Learning*, 8:257–277.

COLLECTIVE BEHAVIOR

Oscillation-enhanced adaptability in the vicinity of a bifurcation: the example of foraging in ants

Eric Bonabeau[1,2,3], François Cogne[1]

(1) CNET Lannion B - RIO / TNT, route de Trégastel, 22301 Lannion Cédex, France
(2) Laboratoire de Physique des Solides, Bâtiment 510, Université Paris-Sud, 91405 Orsay Cédex, France
(3) e-mail: bonabeau@lannion.cnet.fr

Abstract

We suggest that a physical or biological system can be adaptable and respond to a changing environment by maintaining itself in the vicinity of a symmetry-breaking bifurcation. Monte Carlo simulations of a simple probabilistic cellular automaton model of recruitment in ant colonies illustrate how this principle can be implemented with a simple mechanism generating oscillations around the point of instability.

1. Introduction

Adaptability, that is, the ability of a system to reorganize itself to keep up with environmental changes and respond to perturbations, is a characteristic feature of many biological systems and a desirable property of artificial problem-solving devices (Farmer et al., 1986; Huberman & Hogg, 1986). An adaptable system does not get stuck in sub-optimal states, and can switch from one favorable state to the next as often as necessary in a changing environment. We suggest in the present paper that this property can be achieved by means of parametric oscillations in the vicinity of a symmetry-breaking bifurcation. In effect, breaking a symmetry is equivalent to taking a decision or making a choice (Haken, 1984), so that oscillations which constantly drive the system across its critical surface allow new decisions to be taken: if the environment has been modified within the time scale of one oscillation, the system is able to optimize its fitness with respect to the new state of the environment.

We shall not focus here on the mechanisms that produce the oscillations, but rather on how a system can benefit from undergoing oscillations. These oscillations may be generated by more or less complicated internal processes of synchronization, or simply result from an external driving. Let us mention that the interpretation of oscillations in terms of adaptability, although simple and intuitive when cast in physical terms, opens important perspectives in biology, where oscillatory phenomena are widespread: many-body oscillatory phenomena have attracted the attention of many researchers and resulted in the development of various models in which they looked for conditions under which synchronization may appear. Far less attention has been devoted to the adaptive nature of synchronous rythms. In some cases, e.g., collective prey-hunting, the advantage of all individuals or elements of the system being synchronized is obvious; let us also mention the case of neural oscillations, that have been shown to be a plausible candidate substrate for pattern storage, retrieval and recognition. But in many other examples where synchronized oscillations are known to occur, they do so apparently without offering any specific advantage.

The basic theoretical idea is very simple and can be illustrated by assuming that the system under study can be characterized by a behavioral parameter μ and described at the macroscopic (i.e., collective) level by some variable M. In a physical system, μ would be related to the temperature, and M would be the order parameter. Let us further assume that the average dynamical evolution of M is given by the following equation

$$\partial_t M = (\mu - \mu_c)M - M^3. \qquad (1)$$

Although eq. (1) may appear arbitrary, we use it here because it is the simplest normal form exhibiting a critical bifurcation at $\mu=\mu_c$. When $\mu<\mu_c$, the only stationary solution of this equation is M=0 ("disordered phase"), which becomes unstable at $\mu=\mu_c$, and there are two solutions $M = \pm(\mu - \mu_c)^{1/2}$ for $\mu>\mu_c$ ("ordered phase"). This situation is depicted in Fig. 1, where dashed lines represent unstable solutions. Assume that $\mu>\mu_c$ (when $\mu<\mu_c$, M=0, and there is no structure in the system's behavior), and that the system lies, e.g., on the lower branch of solutions (M<0). The addition of an "environmental" (external) field h usually transforms eq. (1) into a normal form exhibiting an imperfect bifurcation

$$\partial_t M = (\mu - \mu_c)M - M^3 + h. \qquad (2)$$

The corresponding bifurcation diagram is represented in Fig. 2, with h>0. If it is more favorable for the system to have a value of M which has the same sign as h, then our system cannot move to a more favorable state (M>0) if it

was previously too deeply in the ordered phase, on the branch with M<0. The system is able to move to the better solution if it is close enough to the point of instability ($\mu=\mu_c$).

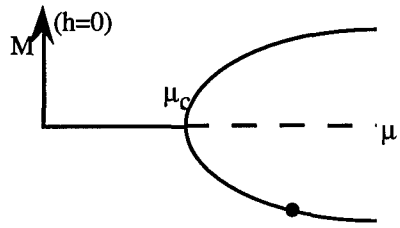


Figure 1: Bifurcation diagram of eq. (1).

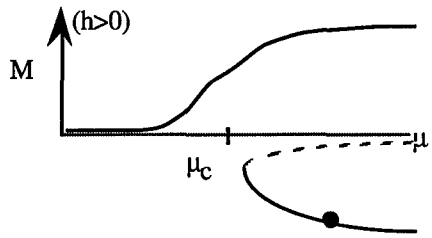


Figure 2: Bifurcation diagram of eq. (2).

Another, somewhat more robust, possibility consists in allowing the system to oscillate around the value $\mu=\mu_c$ more or less regularly in time. In effect, as can be seen in Fig. 3, the better solution can be found by going (slowly enough to enable the equilibrium value of M to be reached) from $\mu>\mu_c$ to $\mu<\mu_c$ (from A to B) and then back to $\mu>\mu_c$ (from B to C). Oscillatory mechanisms therefore reset the system's behavioral parameter, allowing new solutions to be found: this may constitute another reason why oscillations are so widespread in biological systems.

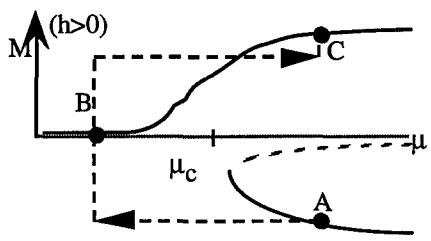


Figure 3: Effect of one oscillation in the behavioral parameter μ on the state of the system.

2. Foraging activities in ants

In order to investigate the effects of sustained oscillations on adaptability, we have developed a model of foraging (i.e. food retrieval) in ant colonies (see, for example, Deneubourg & Goss, 1989; Edelstein-Keshet, 1994; Edelstein-Keshet et al., 1995; Millonas, 1992). A lot of species have a trail-laying trail-following behavior when foraging: individual ants deposit a diffusing chemical substance called pheromone as they return from the food source to the nest, and foragers follow pheromone trails. The process whereby an ant is influenced towards a food source by another ant or by a chemical trail is called recruitment, and recruitment based solely on chemical trails is called mass recruitment.

Because of the pheromone's finite lifetime, there must be a sufficient number of individuals and a low enough error level in trail following for well-defined trails to exist, especially if the path to the source is long. Marked trails allow for more efficient foraging in a given environment. The experimental setup we wish to model comprises a bridge with two branches of lengths L_1 and L_2 separating the nest from a food source (Fig. 4) (notice that since the ants can only follow strictly defined paths — the bridge's branches, the error level is maintained at a low value).

Each ant can choose between branch 1 and branch 2 to go to the food source. This choice depends on the respective amounts of pheromone on the two branches. Both branches are initially selected with equal probability by exploring ants leaving the nest. But ants that use the shorter branch to go to and return from the food source arrive at the nest first: the pheromone they have deposited thus influences other exploring ants to choose the shorter branch rather than the longer one. In this way, the colony optimizes its path to the food source through the amplification of initial fluctuations (self-organizing process).

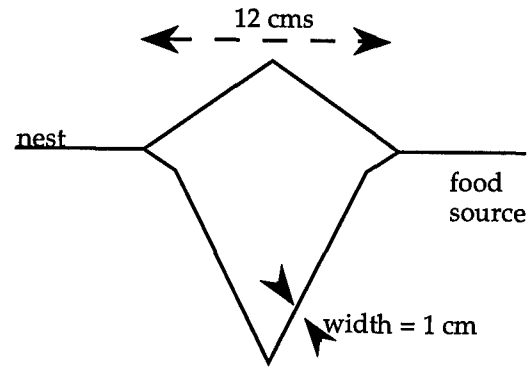


Figure 4: Experimental setup.

However, if the shorter branch is presented to the colony a sufficiently long time after the longer branch, the strong trail established by mass-recruiting ants on the latter prevents the former from being selected: the colony is "trapped" on the longer branch eventhough a better solution is available. One possibility for the colony to find this better solution is to allow

- (partial) trail evaporation on the longer branch, and
- trail amplification on the shorter branch.

This can be achieved in several ways: either with a high enough error level in branch selection (but ants resorting to mass recruitment are usually quite good at following strong trails), or with a variable number of active foragers, since a decrease in this number reduces in theory the efficiency of trail maintenance. This latter mechnism can be implemented through the synchronization of foragers, that is, rythmical patterns in the number of active foragers. The present paper is aimed at showing that this mechanism can indeed endow a colony with some flexibility. In the next section, we introduce a simple model describing the double bridge experiment with a fixed number of foragers, and extend this model to include a variable number of foragers in section 3. Results are discussed in section 4.

The model presented in this paper is not intended to be an accurate model of foraging and oscillatory phenomena in ants, although it is not disconnected from biological reality: some parameters used here may not be completely plausible (especially the relative timescales of pheromone decay, of food decay, and of variations in the number of foragers). The model is rather aimed at illustrating how oscillations may increase adaptability. In fact, we should speak of flexibility rather than adaptability, since flexibility may not always be appropriate: this crucially depends on the colony's environment. As we shall see, the oscillatory behavior may in some cases be "optimal" from the viewpoint of the ratio (food intake)/(energy spent). But such a behavior can be non-adaptive if, e.g., predators and competition with neighboring colonies are taken into account, since oscillations may prevent the colony from being efficiently protected. Finally, we do not treat the full spatiotemporal complexity of foraging activities, since it is not crucial to the understanding of how oscillations enhance flexibility.

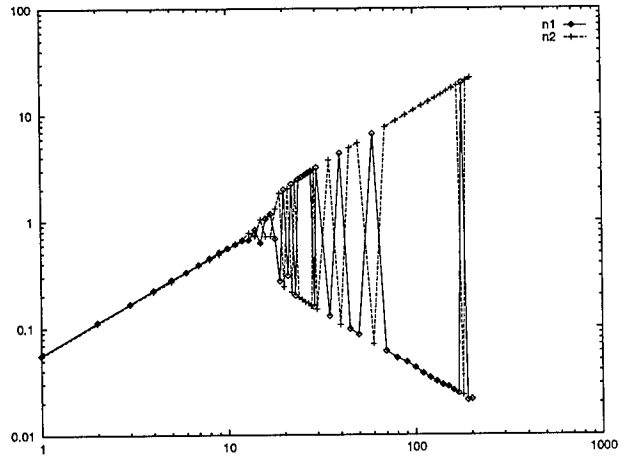


Figure 5: $\langle n_1 \rangle$ and $\langle n_2 \rangle$ for N fixed, $L_1 = L_2$ and $t_1 = t_2$, where t_i is the time at which branch i is presented to the system. Parameters used in the simulations: χ=0.05, $\eta = 1$, $\delta X = 1$, $X_1^{t=0} = X_2^{t=0} = 0$.

3. Model A: fixed number of available foragers

We now introduce a probabilistic cellular automaton (PCA) model of foraging. Let N be the number of available foragers in the colony. We assume for the moment that N is constant. We represent every individual by an automaton which can take three states:

- state S_0 (at nest),
- state S_1 (foraging using branch 1),
- and state S_2 (foraging using branch 2).

The transition probabilities $T_{i \to j}$ from state i to state j are defined as follows. The probability to go from S_i (i=1 or 2) to S_0 is inversely proportional to the length L_i of the branch used to go back to the nest:

$$T_{0 \to i} \propto \frac{(20 + X_i)^2}{(20 + X_1)^2 + (20 + X_2)^2} \qquad (3)$$

All other transition probabilities are equal to zero. Each time there is a transition from S_i (i=1 or 2) to S_0, X_i is incremented by δX: this illustrates the fact that returning ants deposit δX units of pheromone on the branch they use. One time step corresponds to a global update of all automata. Owing to pheromone diffusion and evaporation, X_i decays at a rate χ:

$$X_i^{t+1} = X_i^t (1 - \chi). \qquad (4)$$

Note that X_1 and X_2 play the roles of two spatially uniform, but time varying, external, environmental fields, whereas $\kappa = (1 - (L_2/L_1))$ is a field that is both spatially and temporally constant. Let us test the following three experimental situations:

(1) $L_1 = L_2$ and both branches are present at t=0,

(2) $L_1 > L_2$ ($L_1 = 7.5 = 1.5 L_2$) and both branches are present at t=0,

(3) $L_1 > L_2$ ($L_1 = 7.5 = 1.5 L_2$) and branch 2 is presented to the colony after branch 1, when a stationary regime with branch 1 alone has been reached.

Let N_0, N_1, and N_2 be the numbers of automata in states S_0, S_1 and S_2, respectively. Let us also define $n_i = N_i / L_i$ (i=1, 2), the average number of crossings per

unit time on branch i; n_i is in some cases a more meaningful variable than N_i, because automata in state S_1 may stay longer in this state if $L_1 > L_2$ (it then takes more time to cross branch 1), so that temporal averages might give the wrong impression that more individuals are in state S_1, while in fact more *crossings* occur on branch 2.

As can be seen on Figs. 5, 6, and 7 (which show results of single simulations and not averages over many simulations), the number $N \equiv N_0 + N_1 + N_2$ of available foragers plays a crucial role in the collective dynamics: bifurcations, from disordered to structured foraging, take place as N is tuned.

• In case (1) (Fig. 5), a critical bifurcation from random foraging characterized by $\langle n_1 \rangle = \langle n_2 \rangle$, where the brackets indicate temporal averages computed over 9.10^5 steps after the "stationary" state has been reached, to a foraging strategy favoring one of the two branches (1 or 2 with equal probability), takes place at N=17.

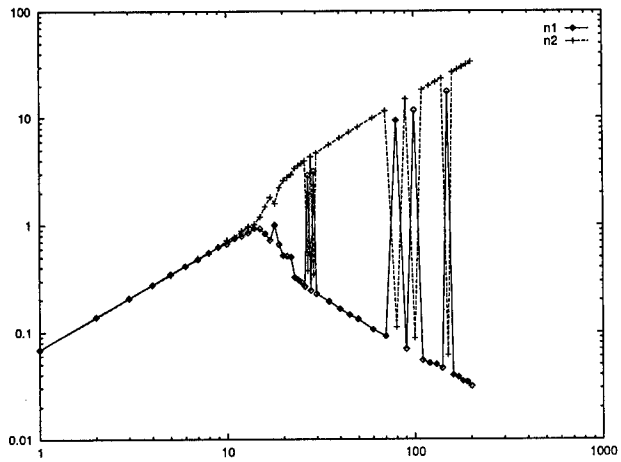


Figure 6: $\langle n_1 \rangle$ and $\langle n_2 \rangle$ for N fixed, $L_1 = 7.5 = 1.5L_2$, and $t_1 = t_2$. Parameters identical to those of Fig. 5.

• In case (2) (Fig. 6), the situation is a little more complicated: there are two successive bifurcations taking place at N_{c1}=11 (critical) and N_{c2}=26 (subcritical) respectively. If $N < N_{c1}$, the path to the source is randomly chosen; if $N_{c1} < N < N_{c2}$, the colony is able to optimize its path to the food source; if $N > N_{c2}$, the colony chooses either one of the branches with a small bias in favor of the shorter branch. This is due to the fact that branch selection takes place by amplification of fluctuations during the first steps of the experiment: once a branch has been selected, it is impossible to switch to the other branch if N is sufficiently large.

• In case (3) (Fig. 7), like in case (2), there are two bifurcations at N_{c1}=13 (critical) and N_{c2}=26 (subcritical). The first one is from random foraging to shorter branch selection (it is very similar to the first bifurcation of case (2)), and the second one from shorter branch selection to longer branch selection. Therefore, for $N > N_{c2}$ the shorter branch is *never* selected when it is presented after the longer one: this is due to the strong autocatalytic selection of the first branch presented, which prevents the second one from being chosen. The amount of pheromone on the longer branch is then too large when the shorter branch is presented for the colony to switch to the more favorable branch. In other words, our model colony with $N > N_{c2}$ is unable to adapt to its new environment, where a shorter branch would allow for a more efficient foraging strategy. Such a non-adaptable behavior is observed in some species of real ants that use mass recruitment (strong selection) for foraging (Beckers et al., 1989; Deneubourg et al., 1986; Deneubourg & Goss, 1989). Let us mention that the size s of the adaptive region (where the colony is able to both adapt and forage efficiently, $N_{c1} < N < N_{c2}$: $s = N_{c2} - N_{c1}$) scales as $s \propto \kappa^{\zeta}$, where $\zeta = 0.15 \pm 0.02$. One way of increasing s is to allow the number N of foragers to vary in time and occasionally fall below the point of instability. To see this from a more "physical" viewpoint, let us define the order parameter $m \equiv \langle n_2 - n_1 \rangle$, and the (iso-N) susceptibility $\chi_N = \partial m / \partial \kappa |_{\kappa \to 0}$. Simulations show that in the vicinity of the critical bifurcation $N = N_{c1}$, the susceptibility diverges: this suggests that highly sensitive choices can be made in the vicinity of the point of instability.

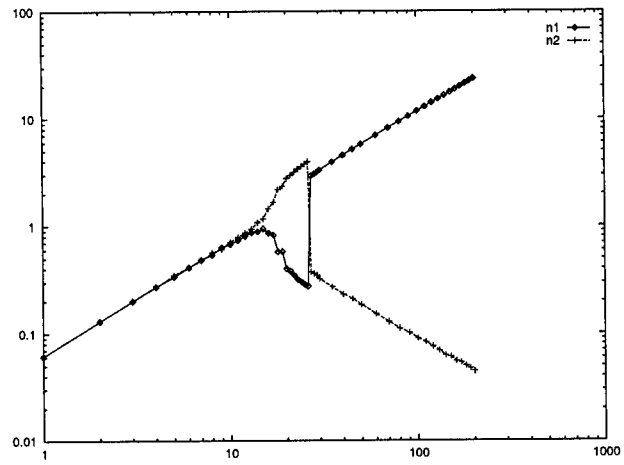


Figure 7: $\langle n_1 \rangle$ and $\langle n_2 \rangle$ for N fixed, $L_1 = 7.5 = 1.5L_2$, and $t_1 < t_2$. Parameters identical to those of Fig. 5.

4. Model B: variable number of available foragers

We have assumed so far that N was constant. In real colonies of ants, N is known to undergo variations in natural conditions depending on external factors, imposed

from outside and independent of the colony's activities, such as temperature or light, or on some more internal factors, such as the degree of starvation, which is known to stimulate foraging activities (Franks et al., 1990; Cole, 1991a,b,c). Oscillations induced by starvation/satisfaction have been observed where the number of active foragers falls to near zero when the colony is satisfied, and rises to more than 200 when it is no longer satisfied. To incorporate the notions of satisfaction and starvation into our PCA, we add a fourth state labeled S_{-1}, which corresponds to a "non-foraging" state. We also introduce the amount of food F available to the colony, which plays the role of a fourth spatially uniform but time dependent external field. Each time there is a transition from S_i (i=1 or 2) to S_0, F is incremented by δF (δF is the amount of food carried back to the nest by one individual). Moreover, F decays because of food consumption: it is reasonable to assume the average total amount of food consumed by n ants to be proportional to n (with some proportionality constant γ), so that here

$$F^{t+1} = F^t - \gamma N_{total}, \tag{5}$$

where

$$N_{total} = N_{-1} + N_0 + N_1 + N_2 \tag{6}$$

is the total number of ants in the colony, including both foraging and non-foraging individuals. A non-foraging individual can either remain in state S_{-1}, or take state S_0:

$$T_{0\to -1} \propto \max\left(\frac{1}{1+e^{-\beta(F-F_0 N_{total})}}, \left(\frac{N_{-1}}{N_{total}}\right)^2 \right) \tag{7}$$

and

$$T_{-1\to 0} \propto \frac{1}{1+e^{\beta(F-\varepsilon)}}, \tag{8}$$

where $F_0 N_{total}$ is an amount of food which exactly meets the colony's needs, β is a positive sensitivity parameter, and ε a small positive constant.

The transition probabilities are now such that too much food stimulates individuals into non-foraging activities, while a lack of food stimulates foraging. For example, in a real nest, specific chemicals emitted by the larvae may indicate that food is needed and therefore stimulate foraging, while foraging individuals stimulate in turn other individuals into foraging activities. (Deneubourg and Franks (1995) developed a model of self-organized collective building in ants that produces oscillations in a similar way: under certain conditions, the nest grows periodically to keep up with population growth, which is a rather slow process; when the population becomes too large, workers start to enlarge the nest with a peak of building activity, so that the nest is enlarged beyond the current needs of the colony, and building can then stop until the population becomes too large again. The number of active builders therefore undergoes periodic oscillations. Deneubourg and Franks (1995) also discuss the adaptivity of such a periodic behavior). The combination of β, F_0 and ε determines the amplitude and period of the oscillations of N, where $N = N_0 + N_1 + N_2$ is the number of active foragers. The N_{-1} non-foraging individuals constitute a reservoir of potential foragers.

Oscillations of N (number of active subsystems) may allow the colony to come back to the point of instability through a modulation of its control parameter N, and to select the shorter branch although it has been presented later. Fig. 8 (where $L_1 = 7.5 = 1.5L_2$) illustrates the ability of the colony to select the more favorable branch in a wider regime. In particular, the shorter branch is always selected as N becomes large, while, it is never selected when N is fixed in the large N limit. Remember that the number of foragers in a real colony is at least about 150-200 when the colony is active, so that the "large" N case is relevant. In order to compare the respective efficiencies of the oscillating N strategy and of the fixed N strategy, we have measured the average amount of food taken back to the nest per unit working time over the duration of the simulation. The result is represented on Fig. 9: we see that the global efficiency of the oscillating system is in general greater, except in a small window. In summary, the fixed N strategy may be *accidentally* more efficient in some particular, *fixed* environment, but is most often more efficient in a changing environment. Since the individuals can take the food back to the nest more rapidly when N varies, they can switch to other activities when there is enough food stored.

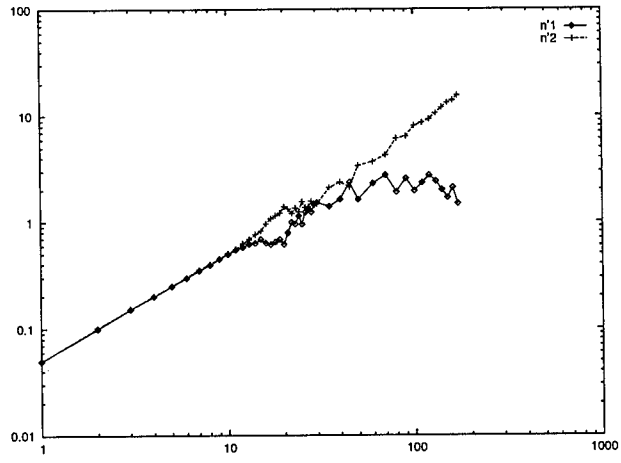


Figure 8: $\langle n_1 \rangle$ and $\langle n_2 \rangle$ for variable N, $L_1 = 7.5 = 1.5L_2$, and $t_1 << t_2$. Parameters used in the simulations: $\gamma = 0.09$, $\chi = 0.05$, $\eta = 1$, $\beta = 10$, $\varepsilon = 0.01$, $F_0 = 90$, $\delta F = 1$, $\delta X = 1$, $X_1^{t=0} = X_2^{t=0} = 0$, $F^{t=0} = 0$.

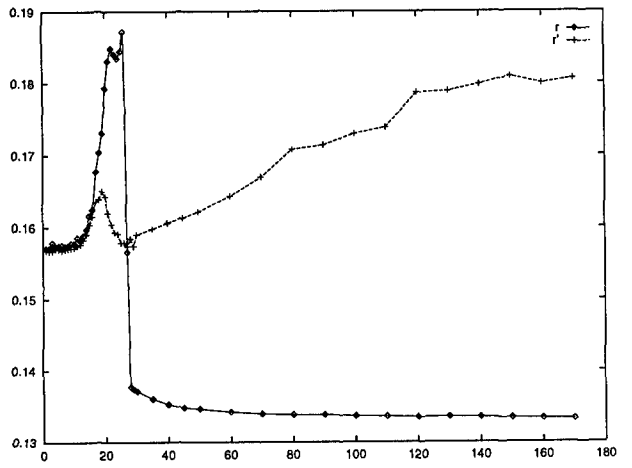


Figure 9: Comparison of the average amount of food taken back to the nest per unit working time with fixed N (r) and with variable N (r'), in the situation where $L_1 = 7.5 = 1.5L_2$ and $t_1 < t_2$.

There is a clear advantage to synchronized bursts of activity, as opposed to a fluctuating control parameter: in effect, synchronization allows the number of active individuals to reach the minimal number required to form a *structured pattern*. If the the number of individuals were fluctuating around an average below the critical number, only random patterns of activity would exist. One may therefore suspect that in some cases, oscillations allow for the formation and temporary maintenance of structures. Once the basic needs of the system have been satisfied, it can switch to another activity. Of course, in nature, the tradeoff between exploration and exploitation must be reasonable: the timescale τ_{osc} of the oscillations should be adapted to the timescale τ_{env} over which the environment is susceptible to be modified, and also to the timescale τ_{sta} needed to reach the stationary regime. Tuning the amplitude of the oscillatory behavior of the system in response to the driving is also important. Moreover, resetting the memory of the system to zero at regular intervals in time may be too drastic a method since some information is lost: for example, in the present case, pheromone trails may completely disappear within the non-foraging part of the oscillation, and this induces an overhead cost, owing to the random exploration required to find a source and reach a stationary state when no trail exists. Therefore, the relaxation timescale τ_{rel} of the system to equilibrium once the dissipative constraints have been switched off also has to be taken into account. Our suggestion is that one should have $\tau_{sta} << \tau_{osc} \approx \tau_{rel} \approx \tau_{env}$ (although this condition was not completely fulfilled in the simulations, since $\tau_{osc} \approx 200 > \tau_{rel} \propto \left(-\ln(1-\chi)\right)^{-1} \approx 20$ (in Monte Carlo steps), we obtained an increased efficiency). We believe that an optimal tradeoff is indeed reached in many natural systems, and that such an adaptive success is certainly the result of evolutionary processes. It is also interesting to note that this mechanism constitutes in some sense an alternative to the division of labor, a phenomenon often observed in social insects (Robinson, 1992), whereby groups of specialized individuals perform different tasks "in parallel", a system usually considered more efficient than if tasks are performed sequentially. But some tasks may require a minimal number of individuals to be performed more efficiently, a constraint that division of labor may prevent from being satisfied.

5. Discussion

The idea that powerful computation or efficient adaptation may take place in a zone of instability has attracted a lot of attention in the last few years (see, e. g., Langton, 1990, 1991; Crutchfield & Young, 1989; Kauffman, 1993): in summary, an optimal tradeoff between exploration and exploitation can be realized in the vicinity of a critical point. Recently, in the context of ant colonies, Solé and Miramontes (1995) have proposed a model in which a perturbation can be coherently transmitted over an extended spatial domain, an ability which is precisely optimal at a critical point, and which possesses a clear adaptive advantage (Adler & Gordon, 1992; Frehland et al., 1985). Bonabeau (1996) showed that some species of ants (resorting to a combination of group and mass recruitment, such as *Tetramorium caespitum*) exhibit a marginally stable behavior that allows them to produce a flexible response when they have to choose one among several food sources of various qualities: their flexibility results from the fact that their behavioral parameters lie very close to a bifurcation point.

One important question that arises if one accepts the idea that computation, coherent spread of information or adaptation are optimized 'at the edge of chaos', is how a system may be capable of maintaining itself at this highly unstable (from the point of view of equilibrium physics) point. Because regions of marginal stability may be small, it is unreasonable to think that biological systems reliably operate in the vicinity of bifurcation points. Our answer in terms of an oscillatory mechanism in the vicinity of the point of instability is one among many potential answers, but one that is biologically plausible. Other possible mechanisms include cases where the control parameter is driven by an external oscillating field (on time scales longer than the driving period, the system is maintained at the bifurcation point under the driving), or simply fluctuations of the control parameter, that are known to lead to noise-induced transitions.

We have described a very simple example where the environment can take only a few states, and where the time dependence of the driving field κ can hardly be made simpler ($\kappa \propto \Theta(t - t_2)$). Complex and rugged fitness or energy landscapes and complex driving processes are certainly more

realistic in a biological setting. Other biological examples relying on the same ideas include the division of labour in insect societies, territoriality in fish (Kortmulder, 1986 ; Goodwin, 1994), or the response to perturbations in the immune system (Bonabeau, 1994). If simple oscillations can enhance the efficiency of a system in a simple environment, we believe that more complex oscillatory mechanisms can have the same result in complex environments (Bonabeau, 1994). More generally, let us assume the existence of a control parameter μ and that a symmetry-breaking bifurcation in the behavior of the system is observed as μ is tuned and crosses a critical surface μ_c. Such a bifurcation can often be associated to a broken symmetry, since the system 'chooses' one among several symmetry-related new branches. Moreover, μ may be 'self-tuned' so as to oscillate around its critical surface. Mean-field evolution equations can be written in the form

$$\partial_t X = F(X,\mu)$$
$$\partial_t \mu = G(X,\mu)$$

where X is the internal state of the system, the control parameter μ is a slow variable often crossing its critical value. Such a mechanism can be related to the notion of self-organized criticality, although, strictly speaking, the system is not *attracted* towards the critical state but rather regularly crosses its critical surface. Sornette (Sornette, 1992) argued along the same lines that most examples of self-organized criticality can be characterized by a feedback mechanism, involving a retroaction of the order parameter on the control parameter which allows the system to maintain itself at the critical state. He further suggested that this feedback mechanism might allow the system to *optimize* its response to a perturbation. Our aim was to make this optimization appear explicitly in a related model.

Oscillatory phenomena on various timescales have been reported in several species of ants such as *Leptothorax acervorum* (Franks et al., 1990; Hemerik et al., 1990), *Leptothorax allardycei* (Cole, 1991a,b,c), *Pheidole hortensis* (Calabi et al., 1983), where the timescale is of the order of 30 minutes, or in *Myrmecina graminicola* and *Messor capitata* (Goss & Deneubourg, 1989) on a larger timescale, of the order of a day or so. The fact that there are not more known instances of synchronized activity in ants certainly stems from the difficulty of discovering such phenomena, which may be much more widespread than current knowledge suggests. The candidate mechanisms that can generate synchronized activity are numerous (e.g., the literature on coupled oscillators is abundant in physics). Moreover, oscillatory phenomena have been found in many biological systems, and it has been argued that oscillations are common in regulatory systems (Berridge & Rapp, 1979): this could be the case precisely because parametric oscillations allow a system to reset its state below a point of symmetry-breaking instability where it can break the symmetry differently. Finally, bursts of synchronized activity that drive the system beyond a critical number allow it to setup *structured patterns of activity*, that could possibly not exist if the system had an average activity below the minimum level for the structures to be formed and maintained.

Perhaps one of the clearest examples of "oscillatory" symmetry-breaking is the case of the playlike territorial behavior of two male fish *Barbus oligolepis* in a tank (Kortmulder, 1986; Kortmulder & Feuth-de Bruijn, 1993). This example is also described by Goodwin (1995) in the same context as the present paper. The observed phenomenon is a sucession of synchronized "heatings and quenches", whereby the aggressiveness of both fishes increases and decreases more or less periodically in time. One occurrence of this behavior results in one part of the tank, say part A, being inhabited by one fish (say fish 1), while part B is inhabited by the other fish (fish 2). When there is a sequence of such behaviors, fish 1 chooses A, then B, and B again, and then A again, etc., with an apparently random pattern of spatial preferences. The other part of the tank is selected by the other fish, "in agreement" with fish 1. It is not possible, on the sole basis of the observations reported in (Kortmulder, 1986; Kortmulder & Feuth-de Bruijn, 1993), to determine the origin of the synchronized behavior of the fish in the tank. But the overall behavior is reminiscent of periodic oscillations, possibly generated by some combination of synchronization processes, that make the system (fish 1 + fish 2) cross its critical surface regularly. Once again, the adaptive advantage of this kind of behavior is to be looked for in the possible modifications of the environment that take place within the time scale of one oscillation.

Acknowledgements

E.B. is greatly indebted to Guy Theraulaz and Jean-Louis Deneubourg for insightful discussions on various biological and theoretical aspects.

References

Adler, F. R. & Gordon, D. M. (1992). Information collection and spread by networks of patrolling ants. *Am. Nat.* **140**, 373.

Beckers, R., Deneubourg, J.-L. & Pasteels, J.-M. (1989). Le recrutement de masse : un exemple de communication trop efficace ? *Actes Coll. Ins. Soc.* **4**, 219-226.

Beckers, R., Deneubourg, J.-L., Goss, S. & Pasteels, J.-M. (1990). Collective decision making through food recruitment. *Ins. Soc.* **37**, 258.

Berridge, M. J. & Rapp, P. E. (1979). A comparative study of the function, mechanisation, and control of cellular oscillators. *J. exp. Biol.* **81**, 217-279.

Bonabeau, E. (1994). Self-reorganizations in a simple model of the immune system. *Physica A* **208**, 336.

Bonabeau, E. (1995). Marginally stable swarms are flexible and efficient. *J. Phys. I France* **6**, 309-320.

Calabi, P., Traniello, J. F. A. & Werner, M. H. (1983). Age polyethism: its occurrence in the ant Pheidole hortensis, and some general considerations. *Psyche* **90**, 395-412.

Camazine, S. (1991). Self-organizing pattern-formation on the combs of honeybee colonies. *Behav. Ecol. Sociobiol.* 28: 61-76.

Cole, B. J. (1991a). Short-term activity cycles in ants: generation of periodicity by worker interaction. *Am. Nat.* **137**, 244.

Cole, B. J. (1991b). Short-term activity cycles in ants: A phase-response curve and phase resetting in worker activity. *J. Ins. Behav.* **4**, 129.

Cole, B. J. (1991c). Is animal behaviour chaotic? Evidence from the activity of ants. *Proc. R. Soc. Lond. B* **244**, 253.

Crutchfield, J. P. & Young, K. (1989). Computation at the onset of chaos. *Phys. Rev. Lett.* **63**, 105-108.

Deneubourg, J.-L. (1977). Application de l'ordre par fluctuations à la description de certaines étapes de la construction du nid chez les termites. *Ins. Soc.* 24: 117-130.

Deneubourg, J.-L., Aron, S., Goss, S., Pasteels, J.-M. & Duerinck, G. (1986). Random behaviour, amplification processes and number of participants: how they contribute to the foraging properties of ants. *Physica D* **22**, 176-186.

Deneubourg, J.-L. & Goss, S. (1989). Collective patterns and decision making. *Ethol. Ecol. & Evol.* **1**, 295-311.

Deneubourg, J.-L. & Franks, N. R. (1995). Collective control without explicit coding: the case of communal nest excavation. *J. Ins. Behav.* **8**, 417-432.

Edelstein-Keshet, L. (1994). Simple models for trail-following behaviour. Trunk trails versus individual foragers. *J. Math. Biol.* **32**, 303-328.

Edelstein-Keshet, L., Watmough, J. & Ermentrout, G. B. (1995). Trail following in ants: individual properties determine population behaviour. *Behav. Ecol. Sociobiol.* **36**, 119-133.

Farmer, J. D., Packard, N. H. & Perelson, A. S. (1986). The immune system, adaptation, and machine learning. *Physica D* **22**, 187-204.

Franks, N. R., Bryant, S., Griffiths, R. & Hemerik, L. (1990). Synchronization of the behaviour within nests of the ant *Leptothorax acervorum* (Fabricius) — I. Discovering the phenomenon and its relation to the level of starvation. *Bull. Math. Biol.* **52**, 597-612.

Frehland, E., Kleutsch, B. & Markl, H. (1985). Modelling a two-dimensional alarm process. *BioSystems* **18**, 197.

Goodwin, B. (1994). *How the leopard changed its spots. The evolution of complexity*. New York: Charles Scribner's Sons.

Goss, S. & Deneubourg, J.-L. (1988). Autocatalysis as a source of synchronized rythmical activity in social insects. *Ins. Soc.* **35**, 310-315.

Haken, H. (1983). *Synergetics*. Berlin: Springer Verlag.

Huberman, B. & Hogg, T. (1986). Complexity and adaptation. *Physica D* **22**, 376-384.

Kauffman, S. A. (1993). *The Origins of Order*. New York: Oxford University Press.

Kortmulder, K. (1986). The congener: a neglected area in the study of behaviour. *Acta Biotheoretica* **35**, 39-67.

Kortmulder, K. & Feuth-de Bruijn, E. (1993). On some generative orders of behaviour. *Acta Biotheoretica* **41**, 329-344.

Langton, C. (1990). Computation at the edge of chaos. *Physica D* **42**, 5.

Langton, C. (1991). Computation at the edge of chaos : phase transitions and emergent computation. PhD Dissertation, University of Michigan, 1991.

Millonas, M. (1992). A Connectionist Type Model of Self-organized Foraging and Emergent Behavior in Ant Swarms. *J. theor. Biol.* **159**, 529-552.

Robinson, G. E. (1992). Regulation of division of labor in insect societies. *Annu. Rev. Entomol.* **37**, 637-665.

Solé, R. V. & Miramontes, O. (1995). Information at the edge of chaos in fluid neural networks. *Physica D* **80**, 171-180.

Sornette, D. (1992). Critical phase transitions made self-organized *J. Physique I (France)* **2**, 2065.

Wilson, E.O. (1975). *Sociobiology*. Cambridge, MA: The Belknap Press of Harvard University Press.

Dominance interactions, spatial dynamics and emergent reciprocity in a virtual world.

Charlotte K. Hemelrijk

AI Lab, Department of Computer Science, University of Zürich, Winterthurerstr 190, CH-8057 Zürich, Switzerland, Fax 0041-1-363 00 35, Email: hemelrij@ifi.unizh.ch.

Abstract

In many studies on social demeanour it is overlooked how complex social behaviour may result from simple self-reinforcing interactions between entities. In this paper an individual-oriented computer model is used to study whether the spatial dynamics between artificial agents and their autocatalytic interactions may result in a rank-related differentiation of social-activity profiles and 'repayment' of social acts. The entities are completely identical at the start of the simulation and very simple: they lack a predefined tendency to reciprocate, but are gregarious and perform interactions in which winning is self-reinforcing. They perceive the others' capacity to win either indirectly, by refering to former experiences with partners, or directly. In this artificial world entities can be ordered according to a dominance hierarchy and a social system with a spatial structure is formed. Furthermore, entities of different rank have their own behavioural profile and patterns of reciprocation emerge. These patterns are particularly apparent in loose groups and among the simplest entities (i.e. those that perceive rank directly). These epiphenomena may also contribute to reciprocation found in real animals such as primates.

1 Introduction

In studies on social behaviour it is commonly assumed that individual complexity lies at the root of intricate social interaction patterns. In primates, for instance, social complexity is attributed to their high intelligence. It is argued by many that the cognitive capacities of primates are especially manifest in the way they regulate their relationships with group-members (Byrne & Whiten, 1988).

An illustrative example is coalition formation, a phenomenon documented for many primate species. Coalitions occur when a third individual C aggressively intervenes in a dominance interaction between two opponents A and B (act 2 in figure 1). C is considered to 'oppose' the one she attacks (A in figure 2) and to 'support' the other (de Waal & Luttrell, 1988) - in this case B. When on a later occasion B 'pays back' by helping C in a fight, reciprocation of support is said to have occurred. Reciprocation of 'support' was measured in macaques and chimpanzees and thought to result from keeping records of acts given and received and a motivation to pay in return; as such, it was considered as an indication for the high intelligence of these species (de Waal & Luttrell, 1988). These authors claim that chimpanzees (but not macaques) also reciprocate 'opposition'. De Waal & Luttrell see this as evidence for the higher intelligence of these apes (compared to macaques).

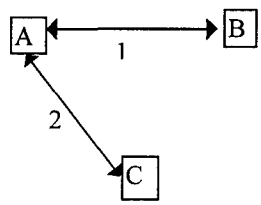


Figure 1. Schematic representation of a triadical interaction: 1) fight between A and B, next 2) C attacks A. The behaviour by C is interpreted as'opposition' against A and 'support' for B.

The terms 'support' and 'opposition' presume that the animals take a specific social context into account, which requires subtle cognitive skills. Such an anthropomorphic approach is common usage in primatology, because complex mental abilities are more easily accepted for monkeys and apes than in any other taxon and are even the main source of interest for many primatologists. The problem, however, is that the terminology already implies an interpretation and that no effort is taken to consider more parsimonious alternatives.

This contrasts strongly with the view adopted by studies in 'New Artificial Intelligence' and 'Artificial Life'. These have demonstrated that behaviour which looks complex and sophisticated to an observer, can often be achieved without central control or global knowledge

(such as keeping records) but instead comes about by simple mechanisms (e.g. Pfeifer & Verschure, 1995). The apparent complexity is then a result of the local interaction of an agent with its environment and with other agents. Along the same lines, Deneubourg and Goss (1989) have shown that apparently 'clever decisions' in ants may emerge from simple self-reinforcing interactions. Therefore, any study trying to explain the complexity of social behaviour, including those on primates, should at the same time ask what part of it must be coded explicitly into the capacities of individuals and what part is determined by the interactions between individuals (Hemelrijk, in press).

To gauge the complexity generating effects of interindividual interactions, I investigate the 'social organisation' of a group of simple, artificial agents by means of an individual-oriented model. By analysing the patterns that emerge, I will supply alternative parsimonious explanations that may help to look at primate behaviour from a new perspective. I will focus on aggressive interactions, dominance, coalitions, and reciprócation, since these are considered pivotal in most primate social organisations.

Whereas social hierarchies are conventionally considered to result from differences between individuals in (possibly inherited) qualities (Ellis, 1994), in the approach adopted here I study dominance ranks that arise by chance and self-reinforcement. This rationale is justified by experimental results from various animal species. These experiments have demonstrated that once an individual has won an interaction (and this may initially be due to chance), this success increases the likelihood of winning again (the so-called 'winner' effect, see Chase et al, 1994).

Using an individual-oriented model formalism, Hogeweg (1988) has shown that the 'winner' effect and the spatial distribution of the engaging entities (so-called SKINNIES) influence each other mutually, resulting in a spatial configuration with dominants in the centre and subordinates at the periphery. This led me to suggest that also reciprocation might result as a side-effect of dominance rank and spatial configuration (Hemelrijk, in press) in the following way: Because higher ranking agents dwell more often in the centre, they will meet others more frequently and therefore also more that are involved in a fight. This leads them to 'support' others in fights more often than do low-ranking ones. In turn, each agent will encounter higher ranking entities more often than lower ranking ones. Consequently, entities will more often 'support' higher ranking agents. According to this scenario, agents 'support' more often those from whom they receive 'support' more frequently in return (but do not necessary return the exact frequency they received) and this is exactly the definition of reciprocation at a group level (see Methods for details). Note that in this way reciprocation comes about without the need to postulate a specific motivation or capacity to 'pay back'.

The aim of this paper is to investigate the interconnections suggested above by studying in a virtual world the behaviour of simple artificial creatures that have a tendency to group. To examine the need for cognition in more detail, two ways to perceive dominance of others, differing in their cognitive complexity, will be compared. In the most simple case, agents (called Perceivers) directly perceive the capacity of winning of those they encounter. In the more sophisticated case, entities (the so-called Estimators) assess dominance of others by recalling their last experiences with partners. Furthermore, the effects of group structure will be studied by comparing social interaction patterns in groups that vary in size and cohesiveness.

2 Methods

In this section I will outline how reciprocity is defined and measured and present a description of the model.

2.1 Operationalisation of reciprocity

At a group level, reciprocity can be approached in two ways, namely according to a model based on acting by one and reacting by another individual (the 'actor-reactor' model) or on acting and receiving by the same individual (the 'actor-receiver' model). In most studies the 'actor-reactor' model is tacitly assumed. It implies that actors direct relatively more acts to those reactors that perform relatively more acts to them in return compared to what these reactors give to other actors. For instance, individual A gives most to the animal that also directs more to A in return than to any other individual in the group. Drawbacks of this model are that complete reciprocation appears impossible for an odd group size and that it is very hard to protect oneself against deception (Hemelrijk, 1990a). For instance, to make sure that the most preferred partner gives more to ego than to others, ego has to trace how often this partner directs acts to all others. In general, individuals therefore have to keep track of all acts directed among all other individuals. To collect such global information is time-consuming and requires extensive cognitive abilities.

Under the 'actor-receiver' model, however, these problems do not arise: here, individuals give relatively more often to those from whom they receive more frequently in return. In this case, the required knowledge is much more 'local', since agents must tune their acts to what they receive from others, but do not have to bother about interactions among others. In addition, complete reciprocation is possible in both even and odd group sizes (Hemelrijk, 1990a).

To test for 'actor-receiver' reciprocation among all pairs of group-members, a specially devised statistic (τ_{Kr}) has been developed (Hemelrijk, 1990ab). This statistic measures the correlation between the corresponding rows of two social interaction matrices and the method reckons with the statistical dependency due to recurrent observations on the same individual (Hubert, 1987). The τ_{Kr}-value for the correlation between a matrix

for 'given' acts and 'received' acts is thus a measure for the degree of reciprocation within a group.

2.2 The Model

The model is individual-oriented and event-driven (see Hogeweg & Hesper 1979; Hogeweg, 1988; Villa, 1992; Judson, 1994). The modelling environment (written in object-pascal, Borland Pascal 7.0) consists of three parts:

* the 'world' (toroid) with its interacting agents,

* its visualisation,

* special entities that collect and analyse data on what happens in the 'world' (cf. the 'recorders' and 'reporters' of Hogeweg, 1988).

The 'world' consists of a regular lattice of 200 by 200 square cells. Each cell can be occupied by only one entity. Conform most primate studies on reciprocation, I will confine myself to small populations of 5-10 individuals. Agents are able to move in one of eight directions. They have an angle of vision of 120 degrees and their maximum perception distance (MaxView) is 50 cells. Agents group and perform dominance interactions according to the sets of rules that will be described below.

2.2.1 Grouping rules

In the primatological literature, two opposing forces affecting group stucture are postulated: on the one hand animals are attracted to one another, because being in a group provides safety. On the other hand, aggregation implies competition for resources and this drives animals apart.
The forces leading to aggregation and spacing are realized in the model by the following set of rules (cf. Hogeweg, 1988; figure 1):

- If an agent sees another within a critical distance (parameter PersSpace), it performs a dominance interaction with that entity. In case several agents are within PersSpace, the interaction partner is chosen at random. If the agent wins the interaction, it moves towards its opponent, otherwise it moves away.

- If nobody is in its PersSpace, but an agent perceives others within a distance of NearView (eight cells), it continues to move on in its original direction.

- If an agent detects others outside NearView, but within its maximum range of vision (= MaxView), it moves towards them.

- If an agent does not perceive any other agent within maxView, it searches for group members by making a turn over an angle (x) at random to the right or left (= SearchAngle).

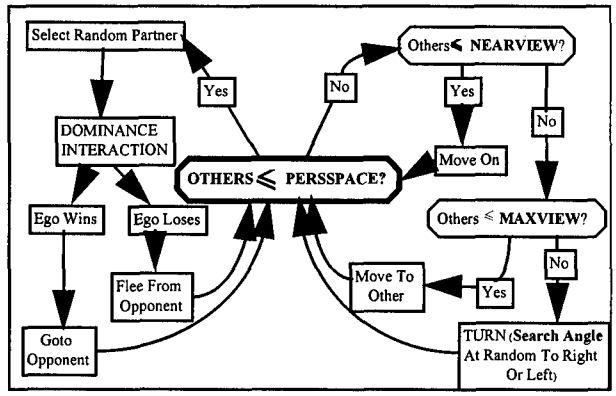


Figure 1. Flow chart for the behavioural rules of the entities

2.2.2 Perception of dominance

A number of hypotheses about how dominance rank is perceived by others are entertained by various authors (Hemelrijk, in press). The most simple one is that the others' capacity to win is directly perceived from external cues, such as pheromones in social insects. In many species, however, dominance may not be recognised externally. The capacities of others may then be estimated on the basis of an individuals' former encounters with a partner. Such a representation asks for more 'cognition' and was used in Hogeweg's SKINNIES (1988). Agents endowed with direct and estimated rank perception will be called Perceivers and Estimators respectively. The effects of both types of dominance perception will be compared in this paper.

2.2.3 Dominance interactions.

Interactions between agents with direct perception of dominance ranks (i.e. Perceivers) are modelled after Hogeweg & Hesper (1983) as follows:

1. Each entity has a variable DOM (representing the capacity to win a hierarchical interaction).

2. After meeting one another in their PersSpace, entities display and observe each others DOM. Subsequent winning and losing is determined as follows by chance and values of DOM:

$$w_i = \begin{cases} 1 & \dfrac{DOM_i}{DOM_i + DOM_j} > RND(0,1) \\ 0 & \text{else} \end{cases} \qquad (1)$$

where w_i is the outcome of a dominance interaction initiated by agent i (1=winning, 0=losing). In other words, if the dominance ratio of the interacting agents is larger than a random number (drawn from a uniform distribution), then agent i wins, else it loses.

3. Updating of the dominance values is done by increasing the dominance value of the winner and decreasing that of the loser:

$$DOM_i := DOM_i + \left(w_i - \frac{DOM_i}{DOM_i + DOM_j} \right) * STPDOM$$
$$DOM_j := DOM_j - \left(w_i - \frac{DOM_i}{DOM_i + DOM_j} \right) * STPDOM \quad (2)$$

The consequence of this system is that it behaves as a damped positive feedback: winning by the higher ranking agent reinforces their relative DOM-values only slightly, whereas winning by the lower ranking gives rise to a relatively large change in DOM. To keep DOM values positive, their minimum value was arbitrarily put at 0.01. STPDOM is a scaling factor and set at 0.5.

4. Winning includes chasing the opponent, who responds by fleeing in the opposite direction (under a small random angle).

In the case of indirect rank perception, the Estimators have to recognise others individually and to remember their personal experience with each partner. Dominance interactions are defined similarly as in the SKINNIES of Hogeweg (1988):

1. If an entity meets another in its PersSpace, it first consults its memory to establish whether it might win or loose a potential dominance interaction with that partner. Hereto, it performs the same dominance interaction as described in (1) and (2), but now based on the mental impressions it has of its own dominance rank and that of the other. If it looses this 'mental battle', it moves away. If it wins, it initiates a 'real' fight. Thus, unlike the Perceivers, the Estimators 'decide' whether or not to attack.

2. If it wins, it 'displays' its expectancy to win as its updated relative dominance rank (=D_i) and the partner displays in return (=D_j). That is:

$$D_i = \frac{DOM_{i,i}}{DOM_{i,i} + DOM_{i,j}}$$
$$D_j = \frac{DOM_{j,j}}{DOM_{j,j} + DOM_{j,i}}$$

3. Winning is decided as in (1), using D_i and D_j instead of DOM_i and DOM_j.

4. Updating of the experiences of each of both entities is done similar to (2), but involves four representations:

$$DOM_{i,i} := DOM_{i,i} + \left(k - \frac{DOM_{i,i}}{DOM_{i,i} + DOM_{i,j}} \right) * STPDOM$$
$$DOM_{i,j} := DOM_{i,j} - \left(k - \frac{DOM_{i,i}}{DOM_{i,i} + DOM_{i,j}} \right) * STPDOM$$

for agent i. Updating for agent j is obtained by replacing $DOM_{i,.}$ by $DOM_{j,.}$.

From now on, the initiation of a dominance interaction will be called 'attack' for short.

2.2.4. Timing regime

Since paralell simulations cannot be run on most computers, a timing regime regulating the sequence of the activations, has to be included. The type of timing regime influences the results of a simulation. A biologically plausible timing regime must be locally controlled, i.e. by other entities and not by a monitor (e.g. Goss & Deneubourg, 1988). In the timing regime used here, each entity draws a random waiting time from a uniform distribution. The entity with the shortest waiting time is activated first. The decay of waiting time is the same for each entity. However, if a dominance interaction occurs within NearView of an agent, the waiting time of this agent is reduced stronger.

2.3 Experimental setup and Data collection

Both Perceivers and Estimators aggregated readily. The values of various parameters were varied to assess their effect on group formation. Especially changes in PersSpace and SearchAngle appeared to influence the cohesiveness of groups. During the runs, this could easily be observed by just looking at the screen: agents either tended to stick together or repeatedly left and joined the main group.

Cohesiveness was measured quantitatively as the mean and the variance of the distance between all entities. In Figure 2 the mean and variance of distance to others are plotted against each other for different values of PersSpace and SearchAngle. Two clusters can be distinguished: the one with low mean and variance corresponds to a case of cohesive grouping, the other represents loose grouping with frequent fissioning and fusion of individuals. Also, mean and variance fluctuate less over time for cohesive than for dispersed groups. Since primate groups vary considerably in cohesiveness, the two classes of groups will be included as a condition in the experimental design described next.

Per type of entity (Perceiver, Estimator) and grouping (Cohesive, Loose) five runs were done for each of three population sizes (N = 5, 8 and 10) giving a total of 60 experiments. After stabilization of the dominance ranks (which typically occurred around 1000*N activations), data were collected for the next 500*N activations. Every 10*N activations the distance between agents was calculated. Dominance interactions were continuously monitored and the following features were recorded: 1) the identity of the attacker and its opponent; 2) their updated DOM-values and 3) cases of triadic interactions that resemble 'support' and 'opposition' (i.e. cases in which a third entity happened to attack one of two agents that were involved in a fight one timestep before).

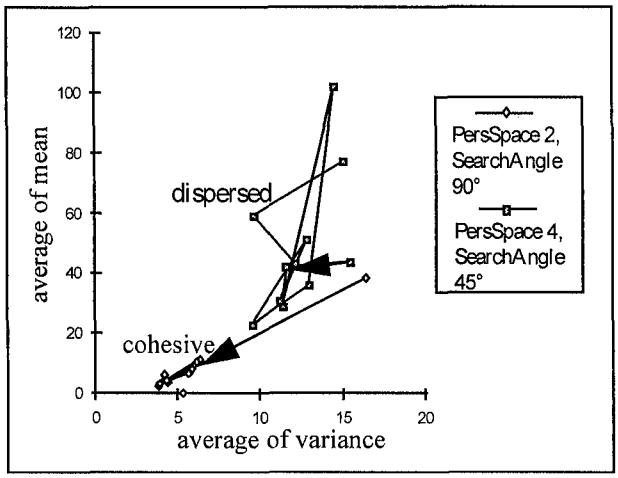


Figure 2. Mean and variance of distance to others averaged over 5 entities with indirect rank perceivement. Arrows indicate the transition from the starting point. Lines connect subsequent points in time.

2.4 Data analysis

To establish whether higher ranking entities indeed attacked and 'supported' others more often, Kendall rank correlations (Siegel, 1956) were performed between dominance rank and the frequency of attack and 'support'. To investigate whether the strength of these associations depended on population size, group type and the type of entity, the correlation coefficients were subjected to a three way analysis of variance.

The degree with which group members reciprocate is expressed by τ_{Kr}-values derived from the Kr matrix correlation (Hemelrijk, 1990ab). To estimate the effects of type of entity, type of grouping, population size and type of behaviour ('support', attack, 'opposition') on the degree of reciprocation, τ_{Kr}-values for these four categories were compared using a four way analysis of variance.

3 Results

3.1 The emergence of dominance hierarchies and correlations with rank.

A dominance hierarchy, such as shown in figure 4, developed among initially completely identical entities in all runs.

Dominance rank appeared to be significantly positively associated with the frequency of attack in 73% of the runs and with 'support' in 40% of the runs. These results ask for a different explanation for Estimators and Perceivers. When Estimators win, they increase their expectancy to win again and therefore initiate increasingly more dominance interactions. Consequently, the number of cases in which a display is directed toward an agent already involved in a conflict itself (i.e. 'support') rises also. Perceivers lack a memory and always attack any other entity they encounter in their persSpace. Therefore, the correlations are completely due to the interplay between dominance interactions and the spatial structure. Because lower ranked entities are more often chased away, they are directed more towards the periphery of the group and hence have less often others in their PersSpace than higher ranking ones have. This reduces their opportunity to attack and 'support' others.

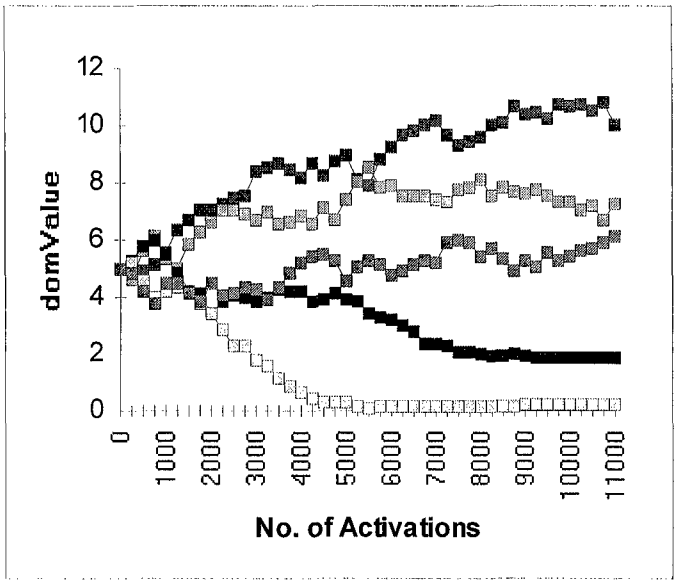


Figure 4. Changes of dominance values over time for 5 Estimators. Stabilization is assumed after 5000 activations

Correlations between dominance and the frequencies of attack are stronger in cohesive than in loose groups (figure 5) and in small than in large groups (data not shown). This holds both for Perceivers and Estimators.

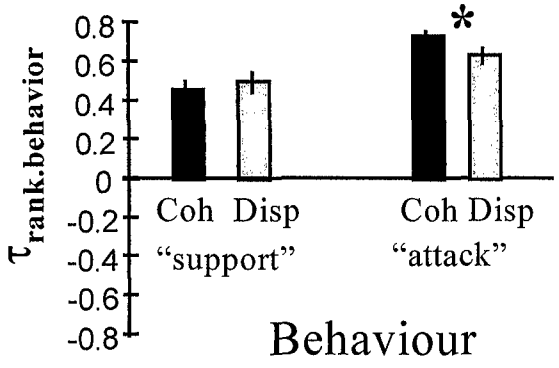


Figure 5. The strength of the correlations between rank and behaviour ('support' and 'attack') represented by their mean τ-values and their standard errors in 30 runs for cohesive groups (Coh) and groups in which agents were dispersed (Disp); the asterisk denotes a significant difference between categories.

Possibly dominance interactions are more equally distributed over group members in cohesive groups than in

aggregations that undergo repeated fissioning and fusion. Similarly, in small groups the interaction distribution may be more homogeneous than in large groups. This may allow for a tighter 'locking' of the dominance hierarchy and consequently stronger correlations between rank and attack.

3.2 Reciprocity

Statistically significant reciprocation of 'support', attack and 'opposition' was observed in respectively 53%, 55% and 18% of the runs.

Reciprocity of 'support' and 'attack' occurred more in loose than in cohesive groups (figure 6) and in large than in small groups (data not shown). This is probably due to the fact that looser as well as larger groups show more persistent sub-grouping. This implies that individuals direct more acts to and receive more acts from members of their own sub-group than that of others, i.e. a coarse form of reciprocity.

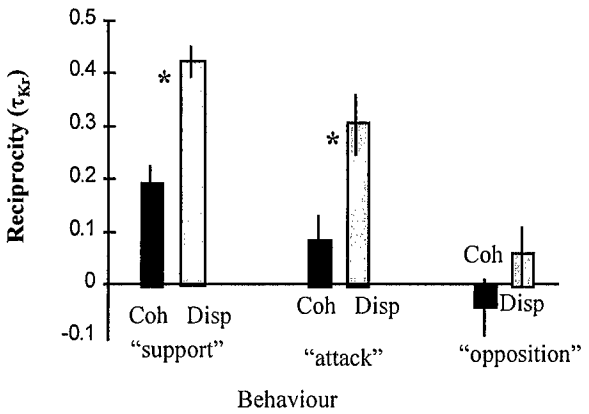


Figure 6. The correlations for the degree of reciprocation of 'support', 'attack' and 'opposition' for cohesive (Coh) and loose (Disp) groupse are represented by their mean τ_{Kr}-values and their standard errors in 30 runs; asterisks denote significant differences between categories.

Reciprocity of 'attack' and 'opposition', but not of 'support' was weaker in the Estimators than in Perceivers (figure 7). This is not surprising, because when Estimators loose from certain partners they are likely to refrain from starting a 'real' fight with the same opponents again, whereas Perceivers attack everybody else independently of former cases of winning or losing. Reciprocity in 'support' is of course not inhibited in Estimators in this way because it is not tied to loosing or winning and reciprocity of 'support' therefore does not differ between both types.

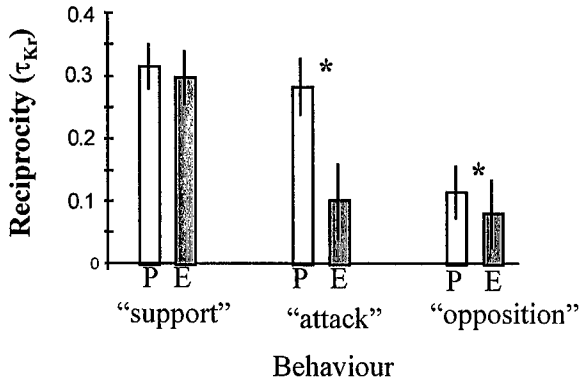


Figure 7. The correlations for the degree of reciprocation of 'support', 'attack' and 'opposition' for Perceivers (P) and Estimators (E) are represented by their mean τ_{Kr}-values and their standard errors in 30 runs; asterisks denote significant differences between categories.

4 Discussion

The model clearly has self-structuring properties. Small parameter changes bring about very different grouping structures. Low values for personal space imply less repulsion among entities and thus more cohesive grouping. A large searching angle has the same effect. The generated group structures in combination with self-reinforcing dominance interactions lead to the emergence of social patterns. These are of a similar complexity as described for real animals: the entities can be ordered in a dominance hierarchy, they behave differently depending on their rank, they take part in ongoing interactions in a way that would be called 'support' in primates and they even reciprocate 'support' and 'opposition'. However, they are able to do this without any of the cognitive capacities that have been postulated for monkeys and apes (such as record keeping and a motivation to pay back). Moreover, these patterns appear to be stronger in the simpler Perceivers than in the more advanced Estimators. This contrasts with the idea of de Waal & Luttrell (1988) who stated that reciprocation of 'opposition' plus 'support' asks for more intelligence than that of 'support' alone.

Interaction processes among these artificial entities may operate in primate groups as well and have similar consequences, but in other aspects these artificial entities of course do not reflect *real* primates at all. In real primates cognitive fundaments may still underly reciprocation, although evidence for this is scarse. The message of this study is that correlations for reciprocity should be controlled for effects of social-spatial structuring before cognitive mechanisms are inferred. An obvious possibility that comes to mind is to cancel out the influence of proximity. However, when this was done (by means of a partial Kr test, see Hemelrijk, 1990b), the outcomes of the model did not change in any significant way. Apparently, the social-spatial structuring is more

complicated than just an effect of proximity. In all probability the involved nonlinear effects cannot be dealt with satisfactory by the usual statistical procedures.
Alternatively, social-spatial reinforcement may be reduced by separating animals from the complete group. Using such an experimental setup (Hemelrijk, 1994), I indeed found indications that macaques 'supported' others more often after having received a beneficial act than not. Although this suggests that the animals have the potential to pay back, the experimental data appeared incompatible with observations collected on the complete group: in the isolated triads individuals 'supported' partners that they never 'supported' in the intact group. It is therefore unclear what such experiments may tell us about behaviour in a large social network (Hemelrijk, in press).

Another contribution of this model to primate studies is that it directly explains how variation in the degree of reciprocation may come about. Such variation has been documented for a number of primate species, (Packer, 1977; Bercovitch, 1988; Hemelrijk & Luteijn, in prep). In line with the results of this study it can be attributed to differences in cohesiveness and group size. Enlarging group size in the virtual world affects social patterning in a similar way as decreasing cohesiveness: in both cases the increased reciprocity is probably due to incremental unevenness in the distribution of interactions.
Field studies in primates have shown that cohesiveness and group size depend considerably on the size and degree of clumping of food sources (van Schaik, 1989). Models on 'artificial apes' of te Boekhorst and Hogeweg (1994ab) have even demonstrated that such effects may override originally assumed species-specific differences in group structure. Up till now I have created cohesive and loose groups by changing parameters that steer the aggregation of agents. In a future version I will extend the model by examining how food distribution affects the emergence of different group types and hence patterns of social interaction.

Acknowledgements

I am grateful to Bernd Goetz and René Schaad for introducing me to object-oriented programming. I like to thank René te Boekhorst for improving former versions of this paper and Rolf Pfeifer for continuous support. This work is supported by the Swiss National Science Foundation, grant number 21-34119.92.

References

Bercovitch, F. B. 1988. Coalitions, cooperation and reproductive tactics among adult baboons. *Animal Behaviour* **36**: 1198-1209.

Boekhorst, I. J. A. te & Hogeweg, P. 1994a. Effects of tree size on travelband formation in orang-utans:data-analysis suggested by a model. In: *Artificial life*. R. A. Brooks and P. Maes, ed., pp. 119-129. Cambridge, Massachusetts: The MIT Press.

Boekhorst, I. J. A. te & Hogeweg, P. 1994b. Self-structuring in artificial 'CHIMPS' offers new hypotheses for male grouping in chimpanzees. *Behaviour* **130**:229-52.

Byrne, R. W. and A. Whiten 1988. *Machiavellian intelligence. Social expertise and the evolution of intellect in monkeys, apes, and humans*. Oxford, Clarendon Press.

Chase, I. D., Bartelomeo, C., Dugatkin, L. A. 1994. Aggressive interactions and inter-contest interval: how long do winners keep winning ? *Anim. Behav.* **48**: 393-400.

Deneubourg, J. L. and S. Goss 1989. Collective patterns and decision-making. *Ethology, ecology and evolution* **1**: 295-311.

Ellis, L., Ed. 1994. *Reproductive and interpersonal aspects of dominance and status*. Social stratification and socioeconomic inequality. Westport, Greenwood publishing group.

Goss, S. and J. L. Deneubourg 1988. Autocatalysis as a source of synchronised rhytmical activity in social insects. *Insectes Sociaux* **35** (3): 310-315.

Hemelrijk, C. K. 1990a. Models of, tests for, reciprocity, unidirectionality and other social interaction patterns at a group level. *Anim. Behav.* **39**: 1013-1029.

Hemelrijk, C. K. 1990b. A matrix partial correlation test used in investigations of reciprocity and other social interaction patterns at a group level. *J. theor. Biol.* **143**: 405-420.

Hemelrijk, C. K. 1994. Support for being groomed in long-tailed macaques, Macaca fascicularis. *Animal Behaviour* **48**: 479-81.

Hemelrijk, C. K. in press. Reciprocation in primates: from complex cognition to self-structuring. In: *The great apes revisited*. T. N. B. McGrew L. Marchant, ed.

Hogeweg, P. 1988. MIRROR beyond MIRROR, Puddles of LIFE. *Artificial life, SFI studies in the sciences of complexity*. Redwood City, California, Adisson-Wesley Publishing Company. 297-316.

Hogeweg, P. and B. Hesper 1979. Heterarchical, selfstructuring simulation systems: concepts and applications in biology. *Methodologies in systems modelling and simulation*. Amsterdam, North-Holland Publ. Co. 221-231.

Hogeweg, P. and B. Hesper 1983. The ontogeny of interaction structure in bumble bee colonies: a MIRROR model. *Behav. Ecol. Sociobiol.* **12**: 271-283.

Hubert, L. J. 1987. *Assignment methods in combinatorial data analysis*. New York and Basel, Marcel Dekker, inc.

Judson, O. P. 1994. The rise of the individual-based model in ecology. *Trends in ecology and evolution* **9**: 9-14

Mantel, N. 1967. The detectiuon of disease clustering and a generalised regression approach. *Cancer Res* **27**: 209-220.

Packer, C. 1977. Reciprocal altruism. *Nature* **265**: 441-443.

Pfeifer, R. and P. Verschure 1995. The challenge of autonomous agents: Pitfalls and how to avoid them. *The artificial life route to artificial intelligence: building embodied, situated agents*. Hillsdale, New Jersey, Lawrence Erlbaum associates. 237-263.

Schaik, van C. P. 1989. The ecology of social relationships amongst female primates. In: *Comparative socioecology, the behavioural ecology of humans and other mammals*. V. Standen and G. R. A. Foley, ed., pp. 195-218. Oxford: Blackwell.

Siegel, S. 1956. *Nonparametric statistics for the behavioral sciences*. New York, McGraw-Hill.

Villa, F. 1992. New computer architectures as tools for ecological thought. *Trends in ecology and evolution* 7: 179-183.

Waal, F. B. M. de and L. M. Luttrell 1988. Mechanisms of social reciprocity in three primate species: symmetrical relationship characteristics or cognition ? *Ethology and Sociobiology* **9**: 101-118.

A Study of Territoriality: The Role of Critical Mass in Adaptive Task Division

Miguel Schneider Fontán and Maja J Matarić

Instituto de Automática Industrial (IAI)
Consejo Superior de Investigaciones Científicas (CSIC)
La Poveda, Arganda del Rey
Madrid 28500
tel: (34-1) 871-1900 fax: (34-1) 871-7050
schneider@iai.csic.es

Volen Center for Complex Systems
Computer Science Department
Brandeis University
Waltham, MA 02254
tel: (617) 736-2708 fax: 736-2741
maja@cs.brandeis.edu

Abstract

This work demonstrates the application of the behavior–based approach to generating ethologically–inspired adaptive foraging using a division of labor into exclusive spatial territories. First, we use fixed group sizes to evaluate and compare the performance of the two types of adaptive solutions. Second, using a collection of experimental robot data, we empirically derive and demonstrate the critical mass for most effective foraging in our domain, and show the decline of performance of the space division strategy with increased group size.

1 Introduction

This paper describes work that applies the behavior–based approach for autonomous agent control to generating ethologically–inspired adaptive foraging. In contrast to foraging methods explored by various researchers to date (see section 3 for an overview), we employ a territorial principle that implements a division of labor into exclusive spatial areas. First, we use fixed group sizes of two, three, and four robots to evaluate and compare the performance of foraging. Second, using a collection of experimental robot data, we empirically derive and demonstrate the critical mass for most effective foraging in our domain. We also show how the performance of the group declines when the described task division strategy is applied to an increased group size, in a fixed–size global workspace.

2 Motivation

Designing and understanding group behavior are complementary and complex problems in artificial intelligence, robotics, and ethology (Brooks 1990*b*, Matarić 1995). Our previous work introduced a methodology for synthesizing a *basis behavior substrate* for generating various behaviors commonly found in nature, including aggregation, dispersion, following, flocking, and foraging (Matarić 1995). This paper extends our work on homogeneous agent groups executing identical control strategies over the entire environment to a somewhat more complex case of employing a spatial division of labor based on an ethologically common organizational principle of *territoriality.*

As we have argued in Matarić (1992*a*) and Matarić (1994*a*), adaptive group behavior is a balance between minimizing interference and maximizing synergy, and interference is the key stumbling block in the way of efficient group interactions. It is correlated with the spatial density of the agents over the lifetime of a task, so various approaches to resource division can be applied to counter its effects. Evolution has produced territoriality as a stable and recurring behavioral pattern across social species. Territoriality produces a physical division of space and all associated resources such as food and shelter (Gould 1982, McFarland 1987).

In this paper, we explore the effects of territoriality on foraging, the prototypical group behavior we have been using for studying social interaction and learning (Matarić 1994*b*). We apply a synthetic approach that consists of implementing ethologically–inspired behaviors on a collection of mobile robots, evaluating their performance on repeated trials, and looking both for 1) stable, robust, and generally useful behaviors we can derive that help synthesis for various robotics applications, and for 2) analogies and similarities to biological behaviors that may help inform analysis of natural phenomena.

We focus on a particular type of territoriality: one existing in a society with equitable spatial territories assigned *a priori.* In this paper, we address the following questions: 1) how does the size of the group affect the

effectiveness of the territorial solution, and 2) how can territories be dynamically re–assigned if members of the group become temporarily or permanently unable to forage?

The rest of the paper overviews related work, then describes our experimental setup, the robots, the spatial assignment of the territories, and the specific foraging experiments. It then presents experimental results and implications on critical mass in this and other types of tasks featuring spatial territorial division of labor.

3 Related Work

The dynamics of group interaction in physical robots have only recently begun to be studied. The review in this section focuses on work on foraging–style collection tasks using either physical robots or realistic simulations implemented with the goal of studying issues of critical mass and control of group behavior.

Matarić (1992*b*) describes early work on minimizing complexity in controlling a collection of robots, and Matarić (1992*a*) outlines the first empirical results with the basis behaviors for control. These served as the foundation for the subsequent demonstrated group behaviors, including following, aggregation, dispersion, homing, flocking, and foraging (Matarić 1994*a*, Matarić 1995). The work described here used the same behavior–based methodology, constructing the individual agent controllers out of collections of distributed behavior networks. Unlike our previous work, however, which focused on fully homogeneous systems, this paper addresses issues of task division and thus specialization as an approach to interference minimization.

Arkin (1992) describes a simulated foraging task with communication–free robots. In a paper more related to the work described here, Arkin, Balch & Nitz (1993) present simulation work with the purpose of studying the issues of critical mass in a multi–agent retrieval task. Unlike our work, the robots are fully homogeneous, and no strict or dynamic territorial division is implemented. However, complementary results regarding critical numbers of agents for the given task are derived. Both their and our pursuit are motivated at least in part by the search for interactive, social strategies that demonstrate linear and superlinear performance improvements through collective strategies. Arkin & Hobbs (1993) describe the general schema–based control architecture, which bears some fundamental similarities to behavior–based control, and give the critical mass experiments. Finally, Arkin & Ali (1994) present a series of simulation results on related spatial tasks such as foraging, grazing, and herding.

Parker (1992) and Parker (1994) demonstrate work related to ours, on the same set of R2 robots with *a priori* hard–wired heterogeneous capabilities. A toxic waste cleanup task used in the work is equivalent to foraging. The Alliance architecture is proposed as an approach to dynamically assigning tasks to members of a robot groups, and is demonstrated on a group of robots dividing the cleanup task. Unlike the spatial division we employed, Parker (1994) describes a temporal division that sends one robot to survey and measure the environment, then return and report to the rest of the group, which uses the information to then clean up the spill. The approach is complementary to ours and demonstrates a clean tradeoff between spatial and temporal task division, both of which are ubiquitous in nature. The same work also describes a simulated office garbage–collection task in which the division of labor is performed based on a first–come first–served basis using a simple and elegant conflict–resolution scheme. This work shares a common philosophy with ours regarding dynamic task assignment; the spatial territories we employed can also be dynamically reassigned according to the adapting needs of the group.

Beckers, Holland & Deneubourg (1994) describe a fully stigmergic approach to the foraging task. Their group of five robots uses no external sensing or communication to collect the pucks. Instead, through a careful combination of the mechanical design of the robots' puck scoops and the simple controller that moves them forward and in reverse, the robots probabilistically move all of the pucks into a single cluster after a certain period of time. The final location of the cluster cannot be determined *a priori* but the collective behavior is highly repeatable and has minimal sensing and computational overhead. The authors also show critical mass effects by comparing task performance on group sizes ranging from one to five robots. These results are complementary to most other work in the field, in that they demonstrate an extreme near–sensorless and totally communication–free approach to foraging. Our approach uses minimal communication in order to ensure the activity of all other robots and thus adapt to any changes in territories. Even this communication could be eliminated and the information could be obtained through direct sensing, but at a relatively high interference cost. Similarly, Parker (1992) employs communication for temporal division of labor in order to solve the task and minimize interference.

Much work has been done in the field of Adaptive Behavior and Artificial Life on simulating insect colonies and studying their dynamics. This work has served as inspiration for robot experiments, and includes Deneubourg, Goss, Pasteels, Fresneau & Lachaud (1987), Deneubourg & Goss (1989), Deneubourg, Goss, Franks, Sendova-Franks, Detrain & Chretien (1990), Deneubourg, Theraulaz & Beckers (1992), Corbara, Drogoul, Fresneau & Lalande (1993), Colorni, Dorigo & Maniezzo (1992), Drogoul, Ferber, Corbara & Fresneau (1992), and many others.

4 The Experimental Setup

4.1 The Robots

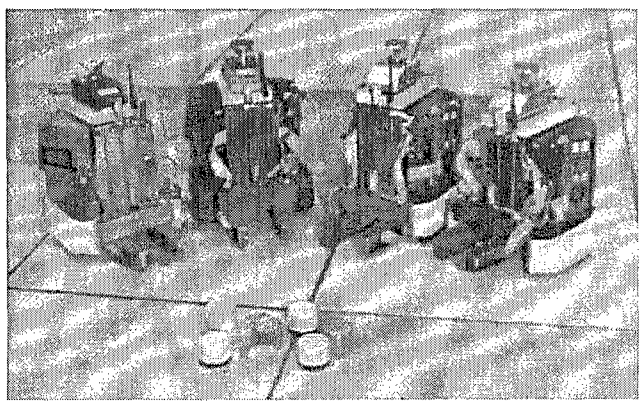


Figure 1: The R2e robot family: the robots are equipped with a differentially–steerable base, a gripper, piezo–electric bump and gripper sensors used to detect collisions and to grasp pucks, infra–red sensors for collision avoidance, and a radio transmitter for absolute positioning and message passing.

The experiments were conducted with a group of four IS Robotics R2e robots programmed in the Behavior Language, based on the Subsumption Architecture (Brooks 1990*a*, Brooks 1986). The robots are fully autonomous and equipped with with on–board power and processing. They consist of a differentially steerable wheeled base and a gripper for grasping and lifting objects. Their sensory capabilities include piezo–electric bump sensors for detecting contact–collisions and monitoring the grasping force on the gripper, and a set of infra–red (IR) sensors for obstacle avoidance and grasping (Figure 1).

The robots are also equipped with radio transceivers, used for determining absolute position and for inter–robot communication. Position information is obtained by triangulating the distance computed from synchronized ultrasound pulses from two fixed beacons, and updated at an approximate rate of 1 Hz. Inter–robot communication consists of broadcasting 6–byte messages at the rate of 1/2 Hz. Once per second, each robot broadcasts a diagnostic "I am alive" message along with its ID number to the whole group. If a robot fails to broadcast for a fixed period of time (in our experiments, 10 seconds), it is considered "dead" by the rest, who consequently adapt their behavior to the new group size and distribution. Hence, each of the robots is given the largest possible group size and can use it to determine how many others are cooperating on the task at each moment.

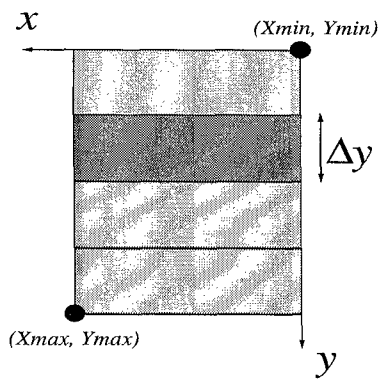


Figure 2: Division of space.

4.2 Definitions

The space ($\Re$) where all the robots work is constrained to a rectangular area defined by the coordinates (X_{min}, Y_{min}) and (X_{max}, Y_{max}) (see Figure 2). All the robots have different working spaces assigned *a priori*, and $\Re$ is divided in rectangular areas with equal height and width. The width of these regions is fixed to $\mid X_{max} - X_{min} \mid$, and independent of the number of working robots. The height (Δy) is dependent on the number of working robots and is defined as:

$$\Delta y = \frac{\mid Y_{max} - Y_{min} \mid}{Working\ Robots}$$

In order to adapt to the dynamically changing group size, each of the robots continually recomputes its adaptive Logical ID, denoted as i. Initially, each robot is given a pre–assigned unique hard–coded ID from the $[0..n]$ interval. During the lifetime of the task, this ID is used to compute the logical ID dynamically, based on the changing group size. Each robot computes its logical ID by determining how many other robots are not sending diagnostic messages, and how many of those have IDs lower than the robot's own ID ($\sigma Dead\ Robots$). Hence, the logical ID i is defined as:

$$i = ID - \sigma Dead\ Robots$$

The logical ID is used to dynamically compute the concepts of the working area, deposit area and deposit point:

- The Working Area (WA) for a robot (R_i) whose logical ID is i, is defined as the region where it looks for pucks, picks them up, and delivers them to the deposit area:

$$WA(R_i) = \{(X_{min}, (i)\Delta y),\ (X_{max}, (i+1)\Delta y)\}$$

- The Deposit Area (DA) is defined as the region to which the robot (R_i) delivers the pucks:

$$DA(R_i) = \{(X_{min}, (i-1)\Delta y),\ (X_{max}, (i)\Delta y)\};\ i > 0$$

$DA(R_i) = \{(X_{min}, Y_{min}), (X_{min}+\delta, Y_{min}+\delta)\}; i = 0$

where δ is the size of the "home" region.

- The Deposit Point (DP) is defined as the point towards which the robot will "home" to deliver the puck:

$$DP(R_i) = (\frac{X_{max}}{2}, (i-1)\Delta y + \frac{\Delta y}{2}); \quad i > 0$$

$$DP(R_i) = (X_{min}, Y_{min}); \quad i = 0$$

As it can be observed from the above definitions, the behavior of the robot whose logical ID is zero ($i = 0$) will be slightly different than that of the rest, as it will be in charge of placing all the pucks it grabs into the "home" region $DA(R_0)$, located in the corner of its work area (see Figure 3).

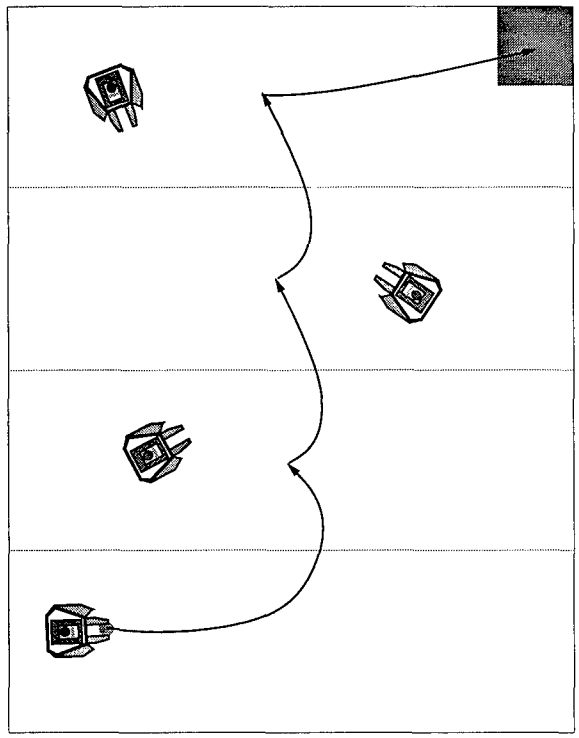


Figure 3: A puck's path from a distant region to the "home" region (upper right corner).

5 Behaviors

The main goal of the group is to collect all the pucks that are distributed over the playground, and take them to the "home" region. In order to reduce interference between robots, each robot is given a working area and a deposit area. Robots look for pucks while in their working area and deliver them once they reach the deposit area. While in the working area, robots actively search for pucks using their finger infra-red sensors, approaching, and picking them up. Once a robot grabs a puck in its working area, it homes to its deposit point and delivers the puck there. Note that each robot has a different deposit point, determined by its logical ID. When the robot crosses the border between its working area and the deposit area, it leaves the puck, and as the robot is now without a puck and outside its working area, it homes back to its working area. While the robot is outside its working area, it does not try to pick up pucks and considers them obstacles. Therefore, the behaviors that use the sensor inputs from the finger IR sensors do not try to approach the pucks, and simply avoid them and any other obstacles that are in the way of the robot on its way back to its own working area.

The robot society utilized in this experiment is a homogeneous group in that all of the members have identical controllers and thus exhibit the same behavioral capabilities. The difference between individuals is only displayed in territoriality, since each confines its behavior to a fixed territory, and can adapt that territory dynamically, depending on the size of the active group.

6 Sources of Interference

As in any multi-agent system, the interaction of a collection of robots produces interference, resulting from a competition for shared resources in the environment, in this case physical space and pucks (Matarić 1994*a*). In the forthcoming sections we outline those aspects that affect the behavior of the system, and their effect on the overall performance of the foraging task.

6.1 Heading Module

Most of the interference between robots arises when they leave their working area and "invade" other areas, either due to sensor and/or effector errors, or through the process of dropping off pucks. In order to minimize the interference while the robots are looking for pucks, they are equipped with a wandering and searching behavior whose goal is to cover the entire working area without invading the neighbors. Since the search area in our environment is rectangular, the wandering behavior always tries to turn in the direction parallel to the longest side of the working area. Consequently, the robots perform a sweep of the area in the direction parallel with the neighbor workarea boundaries, trying to minimize the possibility of invading the neighbor's working area.

The R2e robot family is not equipped with odometric sensors that could provide an estimate of the robot's current position and heading. However, the radio system provides absolute (x,y) position data to each of the robots once per second that can be integrated over time to compute a rough heading estimate. Given two consecutive positions $P_0 = (x_0, y_0)$ and $P_1 = (x_1, y_1)$, the

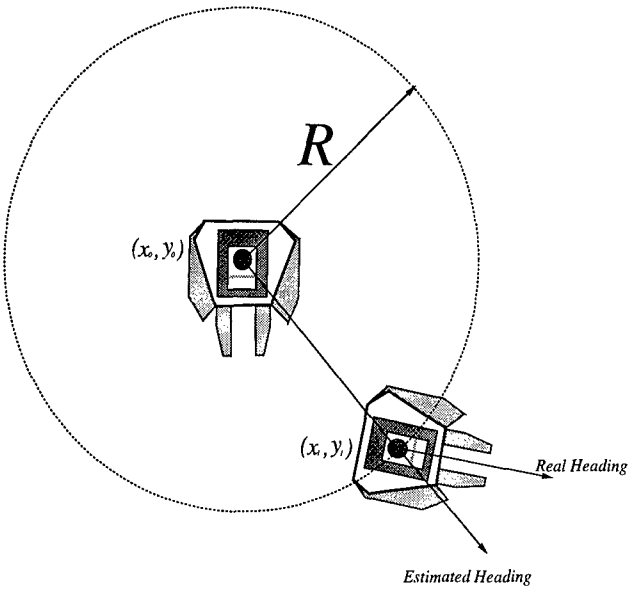


Figure 4: Heading Estimation.

heading of the robot is simply computed as:

$$heading = \arctan\left(\frac{y_1 - y_0}{x_1 - x_0}\right)$$

However, two sources of errors affect the performance of the heading module:

- The robot must move straight and without turns at least by a fixed distance (R) to compute a new heading. However, during the period of time it takes the robot to move from P_0 to P_1, unexpected events, like obstacle avoidance, may occur, that may alter the heading estimate (see Figure 4).
- Errors in the sonar positioning system also affect the heading computation. Over the lifetime of the task, insignificant errors are filtered away, but in small spaces even such errors can dramatically affect the overall achievement of the task. For example, false heading estimation might make the robot "invade" other areas more frequently, and thus increase overall interference.

6.2 Real World Dynamics

Figure 5 illustrates the situation in which a robot leaves a puck in its deposit area and interferes with the robot that is looking for pucks in its own working area. In this case, robot **a** has to home back to its working area, and may interfere with robot **b**.

The size of the working area effects the probability of interference between robots: the smaller the size of the working area the more probable it is that two robots may interfere (see quantitative results in section 7.3). In this

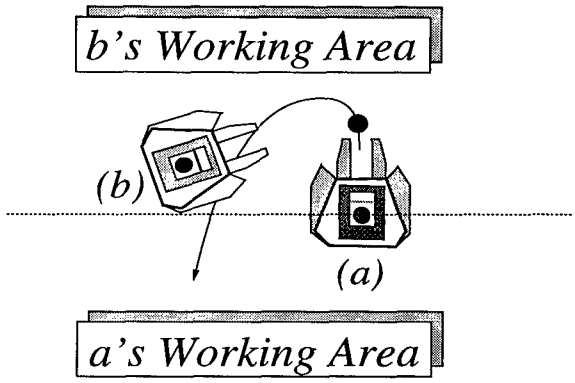


Figure 5: Invasion.

case, interference has side effects in the computation of the heading of both robots: as they are avoiding each other they cannot cover a straight distance (R).

7 Experimental Results

7.1 Experimental Layout

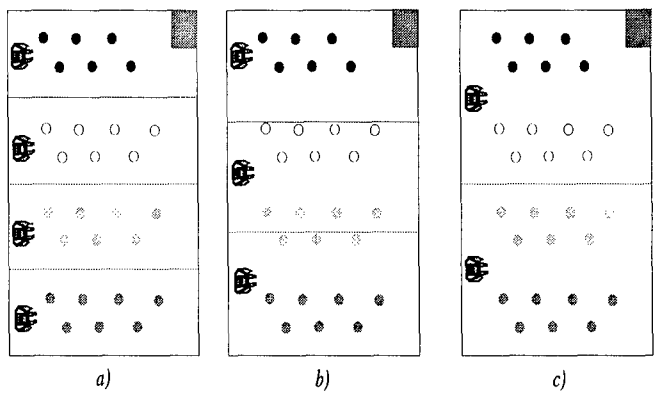


Figure 6: The experimental layout used for three different sets of experiments.

Three different experiments were done in order to quantitatively analyze the performance of the robots in the territorial foraging task. Figure 6 gives an overview of the different scenarios. In situation a), four robots were cooperating to perform the task, and the space was appropriately divided into four equally–sized regions. Robot "0", which is nearest to the home region, had six pucks in its working area, while the rest of the robots had seven. Scenarios b) and c) tested three and two robots, respectively. Initial puck distribution is shown, and differently–colored pucks are used in each of the regions in order to track puck movement over the duration of each trial, using video data gathering, in addition to position data. For each of the different layouts, experiments were repeated five times; all tabulated data show mean values

for each experimental setup. The relative error in the worst case was 8%, measured using standard deviation.

7.2 Data Gathering and Analysis Methods

The following approach was used to compute and analyze the robot data. For each each robot we monitored how long every one of its behaviors was active, and how much time the robot spent in each of the relevant areas around the workspace:

- in the working area $T_{working-area}$,
- outside the working area $T_{outside-working-area}$,
- delivering pucks $T_{delivering}$.

We also gathered two other data points:

1. the number of pucks that had reached the home region 25 minutes after the beginning of the task,
2. the amount of time it took for the robots to place the 80% of the pucks (22 pucks) in the home region.

The total time the robot was working T_{total} could be simply computed as the sum of the times spent in each of the areas:

$$T_{total} = T_{working-area} + T_{outside-working-area} + T_{delivering}$$

Hence, the relevant data point, the percentage of time spent outside the working area (φ) is computed as follows:

$$\varphi = \frac{T_{outside-working-area} \times 100}{T_{total}}$$

This percentage was computed for each of the trials and the final results, shown below, are based on rounded mean values.

7.3 Performance

Number of Working Robots	% of Delivered Pucks
2	60%
3	80%
4	60%

Table 1: Percentage of delivered pucks in 25 minutes.

Time performance in the three different scenarios is shown in Tables 1 and 2. Table 1 indicates the percentage of the total pucks that were successfully delivered to the home region in 25 minutes. Table 2 shows the time (in minutes), the robots needed to deliver the 80% of the total pucks to the goal.

Number of Working Robots	80 % of the Pucks Delivered in
2	32 minutes
3	25 minutes
4	32 minutes

Table 2: Time spent in delivering 80% of the total pucks.

From the observation of these results, it may seem surprising that it took the equally long for two robots, as did four robots, to deliver the same percentage of pucks home. This effect results from the tradeoffs between interference, a bigger search space, and an increased workload per robot. The above described uncertainty in the "heading module" leads to an inaccuracy in the "wandering" behavior, which results in the robot leaving its working area and interfering with the others. The smaller the working area the more dramatically errors in the "heading module" affect the "wandering" behavior, and therefore increasing the probability of a robot leaving its working area and interfering with its neighbors.

A clear relation between the time spent by the robots outside their working area and the time performance can be deduced from the data shown in Table 3. In this table the percentage of the time spent by the robots outside their working area is presented. The results indicate that with narrower working areas robots tend to "invade" other areas more frequently.

4 Robots		3 Robots		2 Robots	
ID	Time	ID	Time	ID	Time
0	26 %	0	20 %	0	8 %
1	29 %	1	24 %	1	8 %
2	29 %	2	18 %		
3	25 %				

Table 3: Percentage of the time spent by the robots outside the working area.

The robots are programmed to avoid any kind of object detected with their lateral IR sensors and their finger–IR sensors while they are outside their working area. However, when obstacles are too close, or difficult to detect (like other moving robots), the bumper–based avoid behavior takes control. Activation of this behavior is a good measurement of interference, because bumper hits are mostly due to inter–robot interference rather than interaction with other objects and boundaries of the environment, which are more likely to be detected with the IRs.

Results shown in Table 4, combined with those shown

Number of Working Robots	Average of Bumper Hits
2	4
3	7
4	20

Table 4: Bumper Hits.

in Tables 1, 2 and 3, highlight the fact that with increased group size the performance of the task division strategy declines due to inter–robot interference, explicitly expressed in terms of bumper hits. As demonstrated, the number of bumper hits grows steeply with increased group size, reflecting inter–robot interference. For example, three robots accrue seven hits while four robots accrue twenty. Three robots perform the task better than two robots (as can be seen in Tables 1 and 2), but interference increases, as is clearly demonstrated in Table 4. Overall, of the two, tree, or four robot sets we tested, three robots are the most efficient choice given the tradeoff between interference and work–load in the particular territorial division.

8 Discussion and Conclusions

Our experimental setup was designed to minimize inter–robot interference, and thus to achieve high task efficiency. This efficiency is directly dependent on the spatial density of the agents, since the most fundamental type of resource competition the robots encounter is that for space. This is a direct result of embodiment, and the underlying cause of basic interactions and behaviors in embodied robots and animals (Matarić 1994*a*). In this particular experiment, and based on the examination of the quantitative data recorded during repeated trials, we have found the sources of uncertainty, inherent to embodied agents, that demonstrate the critical mass for most effective foraging. Furthermore, we postulate that robot data was critical in locating these sources, since simulation results alone would have most likely failed to accurately reproduce the relevant effects, as detailed in section 6.

We have demonstrated that increased group sizes can, in embodied agents, negatively impact the effectiveness of the territorial solution. Intuitively, in a constrained environment, small groups can perform their task better than larger groups simply as a result of the direct relationship between robot density and interference. In future experiments we plan to explore the effects of much lower density on the effectiveness of the group, by significantly increasing the workspace and holding the group size constant.

Matarić (1994*a*) describes a methodology for estimating interference through computing the spatial density of the particular environment–task combination using the robot's physical size, kinematics of movement, and sensory range. This computation must take into account realistic sensory models with appropriate error and uncertainty. The computed value provides a fixed approximation of an efficient group size for the given workspace, assuming no drastic changes in either. However, the computation must be performed every time the size of the environment changes or one or more robots fail. Thus, the density value is best used as a rough approximation.

A robust, ethologically–inspired solution can be applied to optimize task achievement with or without the use of an estimated optimal density. We have demonstrated an approach in which the individual agent controllers are designed to be independent of group size, and furthermore to be adaptive to changes in group size. The current implementation allows for dynamically expanding and reassigning territorial boundaries among the agents. We are also considering an extension that allows superfluous robots to be removed from the workspace. As shown, given the specific parameters of our physical robots and the size of their workspace, the ideal number of robots for the foraging task is smaller than the complete set of available robots. If this fact cannot be computed before the robots are assigned the tasks, then it would be useful to be able to establish it dynamically.

This work focuses on one approach to minimizing interference: the use of spatial division of the task. Inter–agent interference is minimized as a consequence of spatial isolation of the agents' territories. Note that the agents themselves are not further specialized, but are homogeneous across the group. This approach is neatly complementary to the alternatives of using heterogeneous agents (i.e., employing different types of agents that may or may not be spatially isolated) and temporal isolation (i.e., employing division of labor in time rather than space). Other work by Goldberg & Matarić (1996) demonstrates the heterogeneous agent alternative using an implementation of the foraging task on the same robot family.

We plan to compare and tie our territoriality and critical mass results to task division behavior in animal societies in order to find common principles of group organization. We hope that our results can be informative and useful toward the goal of synthesizing group behavior in robots and that they can be used to inspire analysis of group behavior in nature.

Acknowledgements

Miguel Schneider Fontán's work is partially supported by the Spanish Ministry of Science under project I.Ares: CICYT, TAP 816193-C03-2. Maja Matarić's work is supported by the Office of Naval Research Grant N00014-

95-1-0759 and the National Science Foundation Infrastructure Grant CDA-9512448.

The authors wish to thank Dani Goldberg for his tireless help with the misbehaving robots, Lourdes Agapito for her useful comments on earlier versions of the paper, and the insightful reviewers for constructive suggestions.

References

Arkin, R. C. (1992), 'Cooperation without Communication: Multiagent Schema Based Robot Navigation', *Journal of Robotic Systems.*

Arkin, R. C. & Ali, K. S. (1994), Reactive and Telerobotic Control in Multi-Agent Systems, *in* 'From Animals to Animats: International Conference on Simulation of Adaptive Behavior', MIT Press, pp. 473–478.

Arkin, R. C. & Hobbs, J. D. (1993), Communication and Social Organization in Multi-Agent Systems, *in* 'From Animals to Animats: International Conference on Simulation of Adaptive Behavior', MIT Press, pp. 486–493.

Arkin, R. C., Balch, T. & Nitz, E. (1993), Communication of Behavioral State in Multi-agent Retrieval Tasks, *in* 'IEEE International Conference on Robotics and Automation', IEEE Computer Society Press, Los Alamitos, CA, pp. 588–594.

Beckers, R., Holland, O. E. & Deneubourg, J. L. (1994), From Local Actions to Global Tasks: Stigmergy and Collective Robotics, *in* R. Brooks & P. Maes, eds, 'Artificial Life IV, Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems', MIT Press.

Brooks, R. A. (1986), 'A Robust Layered Control System for a Mobile Robot', *IEEE Journal of Robotics and Automation* **RA-2**, 14–23.

Brooks, R. A. (1990*a*), The Behavior Language; User's Guide, Technical Report AIM-1227, MIT Artificial Intelligence Lab.

Brooks, R. A. (1990*b*), Challenges for Complete Creature Architectures, *in* J. A. Meyer & S. Wilson, eds, 'From Animals to Animats: International Conference on Simulation of Adaptive Behavior', MIT Press.

Colorni, A., Dorigo, M. & Maniezzo, V. (1992), Distributed Optimization by Ant Colonies, *in* F. Varela & P. Bourgine, eds, 'Toward A Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life', MIT Press, pp. 134–142.

Corbara, B., Drogoul, A., Fresneau, D. & Lalande, S. (1993), Simulating the Sociogenesis Process in Ant Colonies with MANTA, *in* 'Toward A Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life', pp. 224–235.

Deneubourg, J.-L. & Goss, S. (1989), Collective Patterns and Decision-Making, *in* 'Ethology, Ecology and Evolution 1', pp. 295–311.

Deneubourg, J. L., Goss, S., Franks, N., Sendova-Franks, A., Detrain, C. & Chretien, L. (1990), The Dynamics of Collective Sorting, *in* 'From Animals to Animats: International Conference on Simulation of Adaptive Behavior', MIT Press, pp. 356–363.

Deneubourg, J. L., Goss, S., Pasteels, J. M., Fresneau, D. & Lachaud, J. P. (1987), 'Self-Organization Mechanisms in Ant Societies, II: Learning in Foraging and Division of Labor', *From Individual to Collective Behavior in Social Insects* **54**, 177–196.

Deneubourg, J. L., Theraulaz, G. & Beckers, R. (1992), Swarm-Made Architectures, *in* F. Varela & P. Bourgine, eds, 'Toward A Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life', MIT Press, pp. 123–133.

Drogoul, A., Ferber, J., Corbara, B. & Fresneau, D. (1992), A Behavioral Simulation Model for the Study of Emergent Social Structures, *in* F. Varela & P. Bourgine, eds, 'Toward A Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life', MIT Press, pp. 161–170.

Goldberg, D. & Matarić, M. J. (1996), Interference as a Guide for Designing Efficient Group Behaviors, Technical Report Computer Science CS-96-186, Brandeis University.

Gould, J. L. (1982), *Ethology; The Mechanisms and Evolution of Behavior*, W. W. Norton & Co., NY.

Matarić, M. J. (1992*a*), Designing Emergent Behaviors: From Local Interactions to Collective Intelligence, *in* J.-A. Meyer, H. Roitblat & S. Wilson, eds, 'From Animals to Animats: International Conference on Simulation of Adaptive Behavior'.

Matarić, M. J. (1992*b*), Minimizing Complexity in Controlling a Collection of Mobile Robots, *in* 'IEEE International Conference on Robotics and Automation', Nice, France, pp. 830–835.

Matarić, M. J. (1994*a*), Interaction and Intelligent Behavior, Technical Report AI-TR-1495, MIT Artificial Intelligence Lab.

Matarić, M. J. (1994*b*), Learning to Behave Socially, *in* D. Cliff, P. Husbands, J.-A. Meyer & S. Wilson, eds, 'From Animals to Animats: International Conference on Simulation of Adaptive Behavior', pp. 453–462.

Matarić, M. J. (1995), 'Designing and Understanding Adaptive Group Behavior', *Adaptive Behavior* **4**(1), 50–81.

McFarland, D. (1987), The Oxford Companion to Animal Behavior, *in* 'Oxford, University Press'.

Parker, L. E. (1992), Adaptive Action Selection for Cooperative Agent Teams, *in* 'From Animals to Animats: International Conference on Simulation of Adaptive Behavior', pp. 442–450.

Parker, L. E. (1994), Heterogeneous Multi–Robot Cooperation, PhD thesis, MIT.

Emergent Adaptive Lexicons

Luc Steels
Artificial Intelligence Laboratory
Vrije Universiteit Brussel
Pleinlaan 2, B-1050 Brussels, Belgium
E-mail: steels@arti.vub.ac.be

Abstract

The paper reports experiments to test the hypothesis that language is an autonomous evolving adaptive system maintained by a group of distributed agents without central control. The experiments show how a coherent lexicon may spontaneously emerge in a group of agents engaged in language games and how a lexicon may adapt to cope with new meanings that arise or new agents that enter the group. The lexicon has several characteristics of natural language lexicons, such as polysemy, synonymy and ambiguity.

Keywords: origins of language, lexicon acquisition, self-organization.

1 Introduction

The origins and evolution of language is still clouded in mystery, despite an extensive literature within linguistics, psychology, anthropology and neurobiology (see a recent overview in [15]). The most common hypothesis being explored in American linguistics is that language is based on a species-specific innate ability (a kind of language organ) and on the refinement of innate knowledge (universal grammar) by a parameter setting process [3]. This hypothesis suggests in turn that the language faculty and universal grammar came into existence due to a series of genetic mutations each giving an adaptive advantage [8] or alternatively that there has been a single 'catastrophic' mutation giving rise to syntax and thus full language [2]. Such proposals are coherent and in principle ammenable to computational experimentation. For example, Batali [1] has investigated whether recurrent neural networks, with prior weights resulting from an evolutionary process, might explain the rapid learning and critical periods found in human language acquisition. Much work remains to be done however to make precise proposals for universal grammar and to show how a simple process of parameter setting may make it possible to acquire all the languages currently found in the world.

Current research on the origins of complexity in general [6], suggests an alternative hypothesis: Language could also be an emergent phenomenon, and this in the two senses of emergence. Language is a 'mass phenomenon' actualised by the different agents interacting with each other. In this sense, language is like a cloud of birds which attains and keeps its coherence based on individual rules enacted by each bird. No single individual has a complete view of a language nor does anyone control a language. The processes underlying language use and language formation become invisible once one focuses on a single 'idealised' speaker-hearer, just like a path of an ant society disappears or becomes incomprehensible when only a single ant is investigated. Second, language might be emergent in the sense that (1) it could spontaneously form itself once the appropriate physiological, psychological and social conditions are satisfied and (2) it could autonomously become more complex, based on the same mechanisms that cause the growth of complexity in other fields [16]: evolution, co-evolution, self-organisation and level formation.

I am exploring this hypothesis in a series of experiments on robotic and software agents that span all aspects of language: grounded meaning creation, lexicon formation, syntax, and emergent phonology. An overview of these experiments is given in [13]. This paper only focuses on lexicon formation. It describes a system that gives rise to adaptive lexicons in a group of distributed agents. The system is discussed here in an abstract fashion, but it has been coupled with meaning creation processes [12], implemented on physical robots, and tested in applications such as spatial vocabulary formation [11]. A more detailed description of the lexicon formation process is found in [10].

Other researchers have been exploring the 'emergent language' hypothesis as well. For example, McLennan [7] and Werner and Dyer [14] have conducted experiments in the origins of communication from an 'alife' perspective. These experiments show that communication arises as a side effect of cooperation if it is beneficial for communication. However these emergent communication systems do not constitute a language in the normal definition of the word. The number of agents is small and fixed. The repertoire of symbols is small (8) and fixed. None of

the properties of a natural language such as hierarchical structure, synonymy, ambiguity, etc. are observed. Most importantly, a defining characteristic of natural language is missing, namely that it is open: At all times a natural language keeps growing and changing to express new and different meanings with a finite but open set of building blocks. These growth and adaptation processes must be sufficiently fast to explain for example that a typical adult has acquired a vocabulary of 100.000 words. Moreover the McLennan and Werner-Dyer experiments are based on genetic evolution as the driving force, which means that the language stays fixed within a single individual, whereas languages clearly evolve within the lifetime of individuals.

Another experiment is discussed by Yanco and Stein [18]. They describe how a communication protocol could emerge in a small group of robots using reinforcement learning. Again, the size of the group involved (2 or 3) and the size of the language (between 2 and 20 words) is small and fixed. A scale up on each of these two dimensions quickly leads to an explosion (12,105,480 iterations for a 20 element language with 3 agents, i.e. 24 hours of processing time in simulation). These combinatorial explosions have also been experienced by other researchers and indicate that the problem of emergent languages is a very difficult one.

The present paper reports a significant advance in the problem how a shared vocabulary could emerge in a group of distributed agents. The advance is partly in speed to reach coherence. But also in terms of the presuppositions that are made, and the complexity of the language generated. More concretely, the results are significant in the following way:

1. The system is open. Agents may enter or leave at all times the population (of course within certain stability bounds). New agents quickly pick up the existing lexicon.

2. New meanings may enter the pool of expressable meanings. The agents will rapidly develop new words for the new meanings (after a few hundreds of interactions).

3. Only those meanings are lexicalised that are relevant from the viewpoint of the environment and the communications that arise.

4. Occasionally incoherences arise, i.e. two agents associate a different meaning with the same word. These incoherences cannot be detected if they do not impact communicative success, but they resolve when it is required to make more fine-grained distinctions.

5. It is not assumed that meanings are uniquely identifiable from the context, as is not the case in normal lexicon acquisition either. Pointing to a brown table and saying "wa" can not only mean 'table' but also 'brown' or 'place to sit'. This uncertainty causes the introduction of ambiguity in the language which gets resolved in further interactions.

6. Context plays a role in disambiguation of a given sentence and may provide information to progressively disambiguate a word (a phenomenon also found in natural lexicon acquisition [4]).

7. Multiple word sentences emerge in order to disambiguate single words.

The rest of the paper is in two parts. Part one presents the proposed mechanism formally and gives some examples. Part two discusses results.

2 The basic mechanism

2.1 Features

We assume a set of agents A where each agent $a \in A$ has a set of features $F_a = \{f_0, ..., f_n\}$. A feature f_i consists of a pair $(a\ v)$ where a is called an attribute and v a value. For example, the agents could have attributes like *weight*, *size*, and *shape*, with respective values $\{oval, round, square\}$, $\{tall, small, medium\}$, or $\{weight, heavy, light, average\}$.

A *distinctive feature set* for identifying an agent a as different from the agents in a group B is a set $D_{a,B} \subset F_a$ such that $\forall b \in B, D_{a,B} \not\subset F_b$. There can be several distinctive feature sets for the same a and B. There can also be none if there is an agent $a_1 \in B$ such that $F_a = F_{a_1}$. For example, assume the following agents and associated feature sets: a_5 : $\{(weight\ heavy)(size\ medium)(shape\ oval)\}$,
$a_4 : \{(weight\ heavy)(size\ small)$
$(shape\ oval)\}$,
$a_3 : \{(weight\ heavy)(size\ medium)(shape\ oval)\}$,
$a_2 : \{(weight\ light)$
$(size\ tall)(shape\ oval)\}$,
$a_1 : \{(weight\ light)(size\ tall)(shape\ oval)\}$.
Then $\{(size\ small)\}$ is a distinctive feature set for distinguishing a_4 from $\{a_5, a_1, a_3\}$. $\{(size\ tall)\}$ and $\{(weight\ light)\}$ are two possible distinctive feature sets for distinguishing a_2 from $\{a_3, a_5, a_4\}$. But there is no way to distinguish a_5 from $\{a_3$.

A set of features K can be used to filter a set of agents M resulting in a subset $C_{K,M} = \{a | K \subset F_a\}$. For example, given the set $M = \{a_5, a_1, a_3, a_4\}$ and $K = \{(size\ small)\}$ then $C_{K,M} = \{a_4\}$.

2.2 The lexicon

A word is a sequence of letters drawn from a finite shared alphabet. An utterance is a set of words. In the experiments reported here, word order does not play a role. A lexicon L is a relation between feature sets and words. A

single word can have several associated feature sets and a given feature set can have several associated words. Each agent $a \in A$ is assumed to have a single lexicon L_a which is initially empty. A feature set of a word in L is denoted as $F_{w,L}$. We can then define the following functions:

- *cover*(F, L) defines a set of utterances U such that $\forall u \in U, \{f \mid f \in F_{w,L}$ *and* $w \in u\}$
- *uncover*(u, L) defines a feature set F such that $F = \{f \mid f \in F_{w,L}$ *and* $w \in u\}$.

2.3 Coherence through self-organisation

The lexicon formation process proposed in this paper, assumes that agents have the capability to create new words (by random combinations of letters from the alphabet) and associate it with a feature set. They also have the capability to form a new association between a word and a feature set. This association can be detected in conversations with another agent when both the word is known and a possible meaning (in the sense of feature set) can be derived from the context. However due to the generative capacity of each agent this adoption of words does not guarantee coherence because different subgroups of agents (in the extreme case each agent) may create their own words and associations instead.

Self-organisation in the sense of the spontaneous formation of dissipative structures by the amplification and damping of random fluctuations [9], [5] is a well known mechanism for achieving coherence without central control. It is here applied as follows: Agents randomly couple words to meanings, and engage in communication. They record the success of a particular word-meaning pair and preferably use that in future communications. This establishes a positive feedback mechanism: A word that is used a lot will have a high communicative success and will hence be used even more.

2.4 Language games

A language game involves a dialog between two agents, a speaker and a hearer, within a particular contextual setting which includes other agents. The language game succeeds when the agents manage to identify a particular object (which is in the present case also an agent), further called the topic of the dialog. There are two types of language games. In the first type the topic is first introduced using extra-linguistic means, for example by pointing. Then the speaker identifies the topic again using linguistic means. The hearer can use such a language game to learn part of the language or check whether the right meaning is associated with the right words. Alternatively a language game could contain only verbal communication which is only possible if the language is already sufficiently developed. In this paper, only language games where speaker and hearer already share the topic are considered.

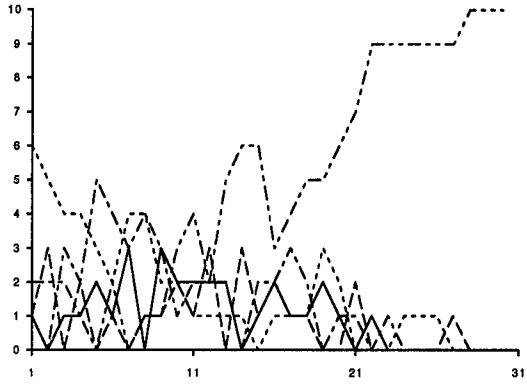


Figure 1: This figure plots the results of an experiment in lexicon formation with 10 agents, 5 possible words, and 1 meaning. It plots the communicative success of each word (y-axis) over time (x-axis). We see a search period in which different words compete to express the same meaning until one gains complete dominance.

The scenario for a typical game is as follows:

1. A speaker and hearer is randomly identified against a background of other agents.
2. The speaker selects another agent which will be the topic and points to the agent so that the hearer shares the topic.
3. Both speaker and hearer identify possible distinctive feature sets that set the topic apart from the background.
4. The speaker selects one set and translates it to words using the cover function.
5. The hearer interprets the utterance using the uncover function and compares it with his expectations.

2.5 Language formation steps

As a side effect of such a language game, various language formation steps now take place. Each of these is discussed in turn.

1. No differentiation possible(step 3 fails)

This situation occurs when the available features are not enough to make a distinction. This stimulates a feature creation process because the available distinctions are not enough to support effective communication. This situation is not further considered in this paper.

2. The speaker does not have a word.(step 4 fails)

In the second case, at least one distinctive feature set S is detected but the speaker s has no word(s) yet to express it. The language game obviously fails. However the speaker may create a new word (with a probability $p_w = 0.05$) and associate it in his lexicon with S. This is for example the case in a dialog printed out as follows:

```
Dialog 102 between a-2 and a-1 about a-4.
```

```
    Context: {a-4 a-1 a-5 a-3 a-2}
    Distinctive: ((size small))
 a-2: ((size small)) -> ? -> a-1: nil
   New word:  a-2: ((size small)) -> (p o)
```

What happened here is the following. a_2 is the speaker, a_1 is the hearer and they try to differentiate a_4 from $\{a_1, a_5, a_3, a_2\}$. There is only one distinctive feature set: $\{(size\ small)\}$. a_2 does not have a word for this feature set and therefore a-1 cannot extract its meaning. a_2 creates a new word. The new word '(p o)' is associated with the set $\{(size\ small)\}$. Newly created words are not directly used but will be in the next conversation.

3. The hearer does not have a word.

In the next case, at least one distinctive feature set S is detected and the speaker s can construct an utterance to express it, i.e. $cover(S, L_s) = W$. However, the hearer does not know the word. Because the hearer has a hypothesis about possible feature sets that might be used, he is able to extend his lexicon to create associations between the word used and each possible feature set. Note that if there is more than one possibility, the hearer cannot disambiguate the word and retains the ambiguity in the lexicon.

4. The speaker and the hearer know the word.

4.1. The meanings are compatible with the situation.

The next situation is one where the speaker and the hearer both manage to formulate distinctive feature sets and know a word covering one or more of them. The speaker produces the word and the feature set uncovered by the hearer is one of the feature sets that are distinctive for the topic. In this case the dialog is a success and both speaker and hearer achieve communicative success. This is for example the case in the following dialog:

```
Dialog 125 between a-2 and a-1 about a-4.
    Context: {a-1 a-3 a-2 a-4 }
    Distictive: ((size small))
 a-2: ((size small)) -> ((p o))
              -> a-1: (((size small)))
```

Note that it is possible that the speaker and the hearer use different feature sets, but because the communication is a success there is no way to know this. Semantic incoherences persist until new distinctions become important and disambiguate them.

4.2. The meanings are not compatible with the situation.

The same situation as before may arise, except that the feature set uncovered by the hearer is not one of the feature sets expected to be distinctive. In this case, there is no communicative success, neither for the speaker or the hearer.

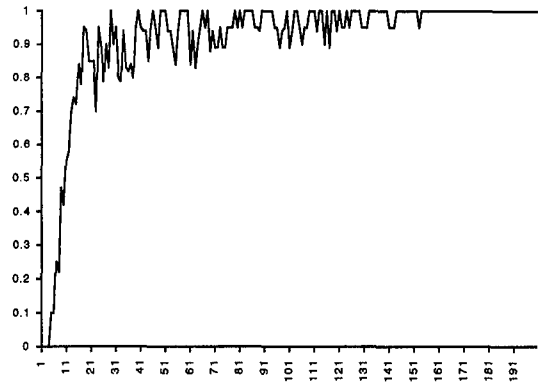


Figure 2: This figure plots the formation of a language from scratch. 4000 language games are shown, involving 5 agents and 10 meanings. The x-axis plots the number of language games (scale 1/20). The y-axis shows the average communicative success.

3 Experimental Results

3.1 One-word utterances

Here are the results of a typical experiment (see fig.2). It starts with the 5 agents given earlier and a dozen features (size, weight, shape) that can be used to differentiate one agent from the others. After a dozen conversations, the first word "(p o)" has been created and is used consistently by all agents to mean *(size medium)*. A dozen conversations later, the word "(f o)" appears to have propagated in the population to mean *(size small)*. Further conversations lead to further increases of the lexicon but also to the first occurrences of ambiguity: the word "(j u)" is used both for *(weight light)* and for *(size tall)*. The reason is that both can be used to distinguish a-2 from the others and an agent who does not know the word yet will retain the ambiguity as seen in the next conversation.

```
Dialog 423 between a-3 and a-1 about a-2.
    Context: {a-3 a-2 a-5}
    Distinctive:
       (((SIZE TALL)) ((WEIGHT LIGHT)))
 a-3: ((SIZE TALL)) -> ((Z U)) <- a-1: nil
   New word: a-1: ((SIZE TALL)) -> (Z U)
   New word: a-1: ((WEIGHT LIGHT)) -> (Z U)
```

When later the word "(z u)" is used to identify the same object within the same context, the communication will succeed and so no disambiguation is possible. Disambiguation only takes place when a-1 uses "(z u)" in a situation where the description *(weight light)* is not appropriate. We often see also a struggle going on between different words competing for the same meaning which will eventually resolve itself in favor of one word as illustrated in fig. 1. For example, some agents use "(j u)" in the same circumstances as others "(z u)". Some agents

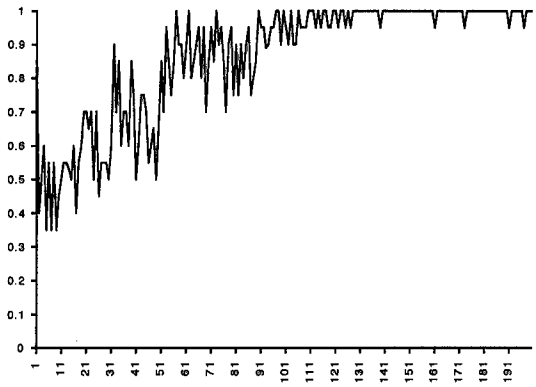


Figure 3: This figure plots the adaptation of the lexicon after six new meanings have been added. Coherence is reached again after 2000 language games.

know about both meanings but preferentially use the one that had the most success in the past. After about 4000 language games the lexicon stabilises as all distinctions that need to be made have been lexicalised.

3.2 Multiple word utterances

When a new feature is introduced, the lexicon gets extended as agents now use it as a distinctive feature (fig 3.). Thus we have introduced color with possible values *blue*, *red*, *yellow* and *white*. The words "(z i)" are adopted for *yellow*, "(d e)" for *blue*, "(k a)" for *white*. The first associations appear where a combination of features are associated with a single word. For example *(size tall) (color white)* is associated with "(z o)".

Multiple word utterances emerge naturally as the set of features expands and the lexicon adapts to cope with it. Here is an example dialog. a_1 identifies himself using the distinctive feature set {(*size tall*)(*color white*)}. Because a_1 uses "(k a)" for (*color white*) and "(v o)" for (*size tall*), a multiple word utterance is made: "((k a) (v o))". a_3 expects either {(*size tall*)(*color white*)} or {(*weight light*)(*color white*)} as possible distinctive feature sets. a_3 understands both words in isolation, takes the union of the feature sets and sees that it fits with the expectations. The language game therefore succeeds.

```
Dialog 2301 between a-1 and a-3 about a-1.
    Context: {a-1 a-2 a-3 a-6 }
    Distinctive:
      (((size tall) (color white))
       ((weight light) (color white)))
a-1: ((size tall) (color white))
       -> ((k a) (v o))
        -> a-3: (((size tall)) ((color white)))
```

Unknown word-meaning pairs can also be guessed by the hearer in multiple word utterances. For example, in the following dialog a_3 knows the meaning of "(k a)" as being *(color white)*, and infers from this that "(f o)" must be either *(size medium)* or *(weight heavy)*. Both are added to his lexicon and later language games will make it clear which one was intended.

```
Dialog 2464 between a-5 and a-6 about a-3.
    Context: {a-1 a-5 a-3 a-2 a-6 }
    Distinctive:
      (((size medium) (color white))
       ((weight heavy) (color white)))
a-5: ((size medium) (color white))
       -> ((k a) (f o)) <- a-6: nil
    New word a-6: ((size medium)) -> (f o)
    New word a-6: ((weight heavy)) -> (f o)
```

3.3 Entrance of new agents

We have already seen in the previous paragraphs that the (distributed) lexicon adapts itself when new features become available to distinguish between different objects. New words are created and sentences become more complex. Another way in which the lexicon formation process can be seen to be adaptive is because new agents may enter at any time in the population. The new agent will gradually take over words already present but is also a new source of novelty, particularly in the present experiments because a new agent means that new distinctions become relevant. Such an experiment has been carried out (fig 4). A new agent a_6 is created and added to the population. The agent has a random assignment of the various features already in use but new distinctions may now have to be lexicalised. After a few hundreds of conversations, a_6 has acquired his own version of most of words used in the group. More interestingly new distinctions become lexicalised, such as *(weight average)* which is expressed as "(m e)". As the lexicon develops, a rich tapestry of meanings and words emerges where no agent shares exactly the same language but the global system nevertheless manages to achieve quasi-total communicative success.

4 Conclusions

This paper reported computational experiments to study the spontaneous formation of a lexicon between distributed agents in a shared environment. It was shown that self-organization is an effective mechanism for achieving coherence and that many properties of natural languages, in particular synonymy, ambiguity and multiple-word sentences, occur as a side effect of the proposed lexicon formation process. Further work is going on to study the global dynamical properties of the proposed mechanism, to test out applications on robotic and software agents, and to link lexicon formation with other linguistic components.

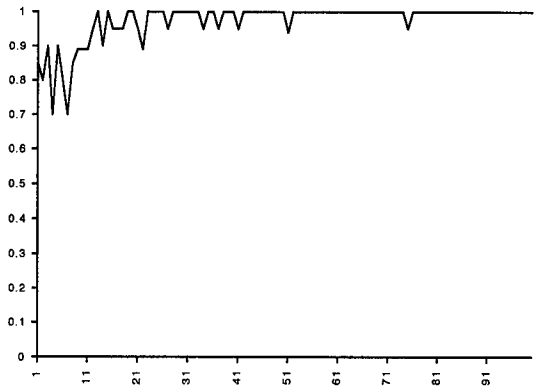


Figure 4: This figure plots 2000 language games illustrating adaptation after a new agent comes in. The x-axis plots the number of language games (scale 1/20). The y-axis shows the communicative success.

5 Acknowledgement

The research and writing of this paper has been financed by the Belgian Federal government FKFO project on emergent functionality (FKFO contract nr. G.0014.95) and the IUAP 'Construct' Project (nr. 20) of the Belgian government, with additional support from the external researcher program of the Sony Computer Science Laboratory in Tokyo.

References

[1] Batali, J. (1994) Innate Biases and Critical Periods: Combining Evolution and Learning in the Acquisition of Syntax. In: Brooks, R and P. Maes (eds.) (1994) Proceedings of Artificial Life IV. The MIT Press, Cambridge MA. p. 160-171.

[2] Bickerton, D. (1995) Language and Human Behavior. University College London Press. London.

[3] Chomsky, N. (1980) Rules and Representations. Brain and Behavior Science. Vol 3. pp. 1-15.

[4] Clark, E. (1990) The Lexicon in Acquisition.

[5] Deneubourg, J-L. (1977) Application de l'ordre par fluctuations a la description de certaines etapes de la construction du nid chez les termites. Insectes Sociaux, Tome 24, 2, p. 117-130.

[6] Langton, C. (ed.) (1995) Artificial Life. An overview. The MIT Press, Cambridge, Ma.

[7] MacLennan, B. (1991) Synthetic Ethology: An Approach to the Study of Communication. In: Langton, C., et.al. (ed.) Artificial Life II. Addison-Wesley Pub. Co. Redwood City, Ca. p. 631-658.

[8] Pinker, S. (1994) The language instinct. Penguin Books. London.

[9] Prigogine, I. and I. Stengers (1984) Order Out of Chaos. Bantam Books, New York.

[10] Steels, L. (1996a) The Spontaneous Self-organization of an Adaptive Language. In: Muggleton, S. (ed.) Machine Intelligence 15. Oxford University Press, Oxford.

[11] Steels, L. (1996b) A self-organizing spatial vocabulary. Artificial Life Journal, **3**(2).

[12] Steels, L. (1996c) Perceptually grounded meaning creation. ICMAS 1996, Kyoto.

[13] Steels, L. (1996d) Synthesising the origins of language and meaning using co-evolution and self-organisation. In: [15]

[14] Werner, G. and M. Dyer (1991) Evolution of Communication in Artificial Organisms. In: Langton, C., et.al. (ed.) Artificial Life II. Addison-Wesley Pub. Co. Redwood City, Ca. p. 659-687.

[15] Hurford, J. (ed.) (1996) Evolution of Human Language. Edinburgh Univ. Press. Edinburgh.

[16] Maynard-Smith, J. and E. Szathmary (1994) The major transitions in evolution. Oxford University Press, Oxford.

[17] Wittgenstein, L. (1974) Philosophical Investigations. Translated by G. Anscombe. Basil Blackwell, Oxford.

[18] Yanco, H. and L. Stein (1993) An Adaptive Communication Protocol for Cooperating Mobile Robots. In: Meyer, J-A, H.L. Roitblat, and S. Wilson (1993) From Animals to Animats 2. Proceedings of the Second International Conference on Simulation of Adaptive Behavior. The MIT Press, Cambridge Ma. p. 478-485

Synthetic Robotic Language Acquisition by Observation

Alexandros Moukas
Autonomous Agents Group,
MIT Media Laboratory,
20 Ames Street, Cambridge,
MA 02139-4307, USA.
moux@media.mit.edu

Gillian Hayes,
Mobile Robots Group,
Dept. of Artificial Intelligence
University of Edinburgh,
EH1 1QL, Scotland, UK.
gmh@aifh.ed.ac.uk

Abstract

Herein we describe work that addresses issues of inter-agent communication skill acquisition by observation in a society of interacting agents. A bee-like society of agents is introduced (most adjacent to these experiments in ethological terms) and an architecture for enabling agents to acquire new skills (learn) by observation is proposed. The architecture is based on a combination of connectionist models, namely Kohonen self-organizing networks and back-propagation networks. A synthetic, movements-based robotic language is devised as a means of verifying the architecture. The architecture enables the agents to learn both how to understand and how to use, by generating new messages, a simple language that can be seen as the agents' symbol grounding of semantic knowledge. The tools used for implementing and testing this work included multiagent simulators and autonomous robotic vehicles as well as other specialized hardware like cameras and transputer boards for vision-based robot positioning.

1 Introduction

In this paper we present work which addresses the issue of inter-agent skill acquisition by observation in a society of interacting agents, and which extends the ideas we introduced in [21]. Infants of many species often gain their skills by observing the behavior of the adult members of the social group. The ability to learn by watching is crucial in enabling agent species to pass on from one generation to the next adaptations to environmental changes that occur on a relatively short timescale.

Since no two embodied agents are identical, skill transfer must be mediated by the perceptions of each individual agent. This is as true for robots as it is for animals: a robot can only learn a skill from another in terms of its *own* sensing and acting capabilities and not by making the assumption that both have identical physical structures.

We have chosen to study the transfer of communication skill between agents and this paper describes a framework within which one robot learns elements of a movements-based language *and what they mean* by observing a teacher robot. The teacher performs a "dance" indicating the presence of a particular type of "food" at a particular distance and bearing. The learner observes the dance and refines its predictions of where the food must be according to the elements of dance it sees by subsequently following the teacher to the food and noting the difference between the predicted and actual locations. At the same time, it learns how to generate the language elements itself given the location of the food source.

Using a language to communicate is clearly beneficial for societies of robots, since information can be communicated succinctly at second hand without each robot having to collect it directly for itself. Having robots learn the language by observation enables them to ground the symbols in terms of their own perceptions and sensorimotor capabilities. The dance elements come to represent the location of food in a way reminiscent of the adaptational role they play in representing the presence of flowers to bees [2].

2 Related Work

Our work is related to that of Yanco and Stein [30, 29], who devised a simple synthetic communication language and implemented it in a simulated and a real environment. The agents they used learned the language by a simple reinforcement learning algorithm; constant reinforcement, positive or negative was used. In the experiments conducted, one agent was the leader and transmitted a series of commands to a varying group of other

agents (followers), describing for them a specific task they should perform (either spin or go forward). A *human* observer determined whether the whole group of followers should be punished or rewarded for their actions.

MacLennan [13] described a synthetic ethology system in which he studied the development of behaviors including communication. His creatures predict other creatures' situations by capturing and understanding broadcast messages from other creatures. The experiments he conducted showed an increased fitness for creatures that utilized this form of global communication.

Weiss [26] focused on action selection and learning in an environment of multiple simulated agents; his agents were reactive, specialized, had only partial knowledge of their environment and used a technique called *Dissolution and Formation of Groups* for addressing reinforcement learning in concurrently active agents.

Mataric [14] worked on a set of composite higher-level behaviors which were based on a set of primitive behaviors and applied them to a group of mobile robots. Using reinforcement learning her robots managed to learn the higher-level behaviors [15]. By introducing progress estimators and heterogeneous reward functions into reinforcement learning, the robots were able to learn to select between various basic behaviors and automatically compose a higher-level behavior [16].

Kuniyoshi implemented a physical system that observed human sequences in an assembly process and transferred the observed skills to a robotic assembly robot [11]. Meuleau introduced a mimetism in an evolving system of car traffic in a simulated road. Although he used a fairly straightforward approach the results obtained showed a significant increase in the performance of the individual cars [19]. Similar (learning from observation) was the work of DeJong [4].

Demiris and Hayes [9] introduced a learning by imitation system where a robotic vehicle learned how to travel in mazes by observing a teacher robot. In their architecture the learning vehicle used a rule-based system to correlate between significant events (ie turn in a maze) and corresponding action taken by the teaching vehicle.

Dautenhahn [3] investigated ways in which robots could recognize others similar to them by keeping bodily contact and matching movements. She went on to investigate how movement patterns might be transferred from one robot to another.

3 Background

3.1 Learning

Learning is more often a transfer of skills than a discovery. Most intelligent species learn and communicate in order to remain competent (by adaptation) in a changing environment.

In order for the infant of many species to behave "rationally" in social terms (that are species-specific) it usually has to acquire new skills. Most of the time, these skills are gained by observing the behavior of the adult members of the social group [24]. When the new skill is a part of its behavioral set it should learn *how* to use it, and in which circumstances. This constitutes its social facilitation [1].

Apart from social facilitation Galef in [7] presented a list of behaviors based on learning by observation: Contagious behavior describes situations in which the behavior of one animal triggers a similar behavior of another animal (for instance the flock formations of birds), whereas in matched dependent behavior the learner observes the reward of the demonstrator/teacher in a specific action, tries to replicate the action, while learning through repetition the triggering conditions of the teacher's behavior. As Galef points out, the examples of learning by observation reported in the ethological literature are legion and their classification into categories supported by the same learning mechanism(s) unknown. Mitchell [20] has given a comparative-developmental definition of imitation in terms of five hierarchical levels of competences an animal must possess; however, this is more a working framework to guide research than a complete explanation of mechanisms.

3.2 Communication and Language

Our work focuses on creating a society of agents that exhibit a form of social behavior. We believe that communication is a prerequisite for cooperation, which is a prerequisite for intelligent social behavior. In this work we are addressing questions like how agents can mediate between the propositional representations of language and the non-propositional representations of perception and action; and how information can be stored and communicated in a situated environment when all agents do not share the same representations.

By introducing communication in the form of a common language in a multi-agent system we enable each agent to conceive and learn about the world in its own terms, while at the same time we facilitate knowledge and skill exchange even among agents with completely different internal representation schemes.

The communication among the agents was designed to be direct (but not directed), one-to-many. Direct communication is a form of communication in which the transmitter is sending information for its own sake, not as a side-effect of one of its actions, as opposed to indirect or stigmetric communication in which information is not transmitted on purpose by the transmitter but instead it is conceived by the unknown number of recipients as a change in their environment [12].

In order for the agents to communicate a common

ground, a *language* must exist among them. A language is independent of its means of transmission and it is not limited to phonemes (spoken) and characters (written). The language proposed here will be communicated via a set of movements in a 2-D space, a *dance*. This language, as all languages, can be formally defined by introducing three concepts. First the *syntax and grammar* of the language denote the rules that the elements of the language should follow to produce meaningful statements. For the purposes of this experiment the syntax of the language is very simple. Because of the simplicity of the proposed language, its syntax could be easily mistaken for the information passed through that syntax; however this is not the case. Those two are kept separate in order to facilitate language extendability. Second the *world knowledge*: in order for a language to be understood, it should describe a domain known to everyone. For example, a person who has been visually impaired since birth could never understand the meaning of words describing colors in an unknown language [31]. In our case the world is known both to the recipient (learner) and to the transmitters of the language since they are situated. Finally the *semantics:* a language needs rules that will link the syntactical structures with the appropriate objects and relations in the world.

A language can be either pre-engineered or developed by the agents themselves; it can also be fixed or adaptable by the robots. Combination of these parameters leads to four different cases, which all have their advantages and disadvantages. For instance a pre-engineered non-adaptive language, while having a very quick start-up time on the agents, may not suit them in their tasks and while easier to learn, the agents will not be able to expand the language when coping with unpredicted circumstances. On the other hand an agent-developed adaptive language while guaranteeing suitability has the worst learning curve [29].

Our exact implementation of the language was heavily influence by the design of the robots we used (turn radius, size, holonomic or not).

*3.3 The Honey Bee (*Apis Melifera*)*

Many species use communication to maximize their survival chances. One example of the use of inner-species communication through movement is that of bees. Honey bees (*Apis Melifera*) communicate via a number of different movements, grouped into dances, which have been extensively studied by ethologists during the 20th century, especialy Karl von Frisch [25].

Bees recognize their food sources (flowers) through their patterns and colors (both in the visible band and in the ultraviolet) and distinguish certain flowers from others using patterns visible only in the UV band [17]. They use various methods for distance measurement including the energy they consume during a particular journey. For navigation they use landmarks as well as the sky's brightness patterns (polarization) and the changes in the magnetic field of the earth [8].

As a means of communication among them bees use a special kind of movement called a "dance"; their scouts use this dance to inform other bees at the hive about a food source. Three types of dance exist, the *waggle dance*, the *round dance* and the *dorsoventral abdominal vibrating dance (DVAV)*.

During the waggle dances the scout-bees use the position of the sun as a landmark and they orient their dance in such a way that the angle between the direction of the sun and the direction of the food is the angle between the vertical and the angle of their dance. The duration of the dance denotes the distance of the food source from the hive. Interestingly different bee species have different notations for the distance. For example in some honeybees one 8-shape movement in the waggle dance denotes a 25 meter distance of the food source from the hive, whereas in African bees this distance is as much as 75 meters [17].

The round dance is the simplest form of dance and does not communicate exact distance or food source direction information. Rather, it informs the worker bees in the hive that a food source exists at close proximity (less than 15m). The DVAV or vibration dance is used mainly on foraging colony activities. Using the vibration dance the bees regulate their foraging patterns according to short and long term food availability [28].

Apart from these dances, which are used for recruiting and communication regarding foraging, bees use cues (direction of flight) which transmit hive information as a side-effect of their behaviors [23]. The language of the bees is adapted through evolution to suit their needs. For instance there is no word for "up" in the bee language as shown by [25] (since there are no flowers in the sky).

4 The Synthetic Language and Its Acquisition

The system proposed here enables the transfer of skills from one agent to others by observation. The system can facilitate both one to one and one to many agent learning. The skill to be acquired is the ability to communicate information about food sources by means of a synthetic movements-based robotic language. This language is ad-hoc and devised for the purposes of this project. Table 1 describes the meaning of the basic language elements, the shapes traced out by an agent on the floor.

The agents' internal representation is described in the agents' own terms. For instance distance is not measured in meters but in the time it takes for the agent to go from the hive to the food source. The angle is also measured by the time the agents need to turn. The food type is the "recharging" power of the specific food in minutes of operation (this will become more evident if auto-

Shape	*Measures*	*Human*	*Agent*
Small Square	Distance	0.33 m	1 sec
Large Square	Distance	1.00 m	3 sec
Triangle	Angle	15°	0.1 sec turn
Rhombus	Food type	I,II,III,IV	60..240 sec

Table 1: Explanation of Basic Language Components (shapes). The first column describes the type of the shape, the second the entity it is measuring, the third the units per shape from the external observer's point of view and the fourth the units per shape from the agents' point of view. For instance two large squares, one small square, three triangles and a rhombus denote a food source at a distance of approximately 2.33m from the hive, at an angle of 45° and a type I food source. From the agent's point of view this corresponds to a right turn of 0.6 seconds, and a forward movement of 7 seconds. When it reaches that position it expects a food type with 60 seconds of recharging power. The smallest unit of distance measurement is one small square, 33 cm, which is very close to the length of the robots (35 cm), and provides good resolution.

matic recharging is implemented or a "virtual recharging manager" is sending commands to each agent and altering their internal representation with respect to the food source type – their "hunger").

The experimental setting includes the agents, a "home region" or "hive" and "food sources". A set of agents exist that already possess knowledge of a specific social behavior, in this case how to interpret and carry out a special set of movements, a *dance*, in order to obtain the position and the type of various food sources. The area that will be described by this dance will be a food area, away from the agents' home area.

The dance describes how to reach the food source in terms of angle and distance from the home region, the "quality" and quantity of the food and the type of food the source contains; this information is expressed through the repetitions of triangles, small and large rectangles and rhombi respectively.

Besides the agents that already possess the knowledge of the interpretation of this specific social behavior, another agent exists, a "new-born" that will eventually have to learn the dance by watching the other agents but without being explicitly taught by them.

5 Architecture

The proposed architecture's main goal is the learning of a specific task from an agent without explicit instruction. The approach described here differs from previous learning architectures in that the agents learn not only to understand the synthetic robotic language but to generate it as well (in other words match the non-propositional representations of perception and action to the propositional representations of the language and vice versa).

The architecture is designed so that it can be very easily (automatically in the future) expanded. The experiments require no human intervention; the society of the agents is fully autonomous. The task of communication skills acquisition (language learning) was not chosen on the basis of building a Natural Language Processing and Natural Language Understanding system but as a method of verifying the performance of the architecture.

5.1 Architecture Overview

The skill acquisition behavior of the agent is outlined in Figure 1. The learning procedure begins with the learner agent observing the movements of the demonstrator agent (teacher). These movements trace out on the floor different shapes with different meanings. Three kinds of shapes exist: rectangles, triangles and rhombi that denote the distance, the orientation and the type of the food source respectively, and the teacher inscribes the appropriate number of each. It pauses after each instance of a shape and before switching from one shape to another.

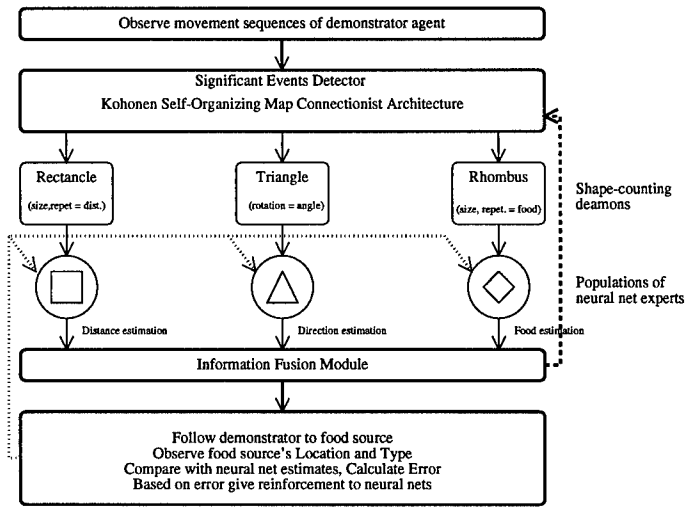


Figure 1: Language Acquisition by Demonstration Architecture Overview

The basic components of the learning system are the following:

1. *Significant Events Detector:* Its role is to distinguish between different types of shapes traced out by the teacher and then to "awaken" the relevant shape-counting daemon. It also detects `end_of_shape` and `new_shape` signals that the demonstrator agent produces during the dance.

2. *Shape-Counting Daemons:* There are three shape-counting daemons, one for each shape, one being active at any time. Each is responsible for counting the repetitions and the size of each shape the demonstrator agent inscribes and reporting those values to the appropriate neural network.

3. *Populations of Neural Networks:* For each shape two back-propagation networks exist. One is used for decoding (converting from the language to the agent's internal representation, used during the training phase) and one for encoding information (converting from the agent's internal representation to the language, used during the demonstration stage).

4. *Information Fusion Module:* This module collects information from the populations of experts, processes it, and forwards it to the *Error Estimators.*

5. *Error Estimators:* After finding the food source, the agent compares the estimates of the neural networks with the actual position and type of the food source and adjusts the weights of the networks accordingly. The weights of both the acquisition and demonstration networks are updated.

5.2 *Significant Events Detector and Kohonen Networks: Shape Detection*

The role of the Significant Events Detector is to infer the shape demonstrated by the teacher agent and detected by the learner (in this particular instantiation via a camera positioned above the teacher). This role extends to the isolation of the important segments of the behavior of the demonstrator agent (with regard to the current task — language learning) from the non-important segments. In other words it helps the agent pay attention only to what's important.

The self-organizing networks were used here as tools for distinguishing between significant events / language components / different shapes in the synthetic movements language communicated between the agents. The input of the Kohonen network is a 20 dimensional vector of 10 pairs of (x,y) coordinates of the robot's position collected at various times within the duration of the "dance" of a single shape. Thus the vector data have both a spatial and a temporal element. The aim of the Kohonen network is to classify the type of shape based on this vector.

More traditional methods for recognizing the inscribed shapes, like implementation of standard vision techniques, were considered. However, preliminary experiments showed that connectionist approaches provided more robust results than the vision approach. Vision algorithms required greater precision in the robots' movements, which are inherently quite inaccurate. Kohonen networks dealt better with the noise of the environment and the precision (or lack thereof) of the robots.

5.3 *Populations of Expert Sub-Agents: Learning the Language*

In order to provide directed feedback and to accelerate the learning process more than one sub-agent is implemented to collectively solve the problem of communication skills acquisition. A popular type of connectionist architecture, a back-propagation artificial neural network is selected for the implementation of a population of expert sub-agents.

Each sub-agent is responsible for a part of the language, a single grammatical entity. When a vehicle communicates a message using the language, the sub-agents of the learning vehicle become active, and vote for their suitability for each language element. The agent that offers the higher bid will take over and correlate the message to an internal representation. If at the end the agent has proven right it will receive positive feedback, otherwise it will be punished (relative to its bid). A closed ecosystem could be devised in the future for elimination of non-competent agents. In case of a new language concept, where no sub-agent seems willing to take over, a new sub-agent would be created.

The critical feature of the neural net populations, that of paired operation, is a result of the nature of the learning process we are interested in: having an agent learn not only a new skill by observing other agents (in this case learning to understand a dance language) but also how to *use* the language later (deferred imitative/observational learning) to *generate* its own messages (as opposed to reproducing the observed messages). Since a straightforward way to extract knowledge from a neural network does not exist, we introduced the following schema: for each component of the language (shape) are two back-propagation networks, one for learning the correlations between the demonstrated dance and the food sources (that enables the agent to navigate to a food source from the home region) and a second for learning the correlations between the food sources and the dance (that enables the agent to generate the proper dance, given a food source).

The first network decodes the information found in the message of the demonstrator from a series of shapes to a meaningful set of data that will allow the agent to safely navigate to the food source from the home region. In other words the agent will have the world knowledge and the syntax of the language and will try to infer its semantics. The second network does the opposite: it converts the information of a food source position into an understandable series of movements. The output of the first network is the input of the second. The food source information is meaningful to the agent alone, because it represents how *it* discovered the food source.

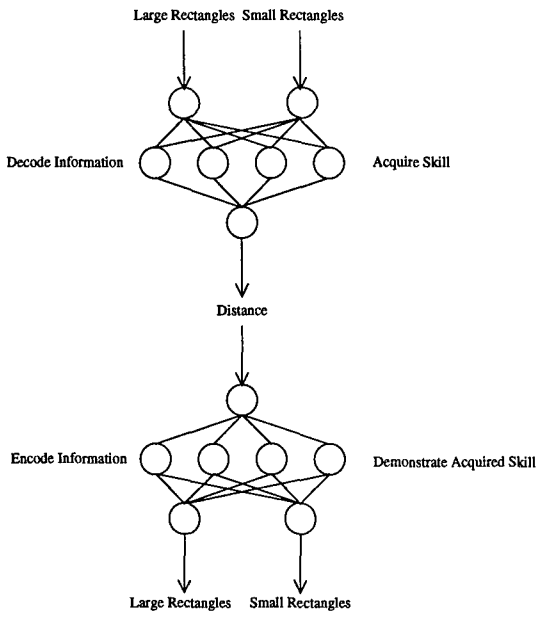


Figure 2: Neural Network Pairs Layout

This information is local because it is observed from the agent's *own* point of view. In order to be understood by the other agents it has to be encoded (transformed) into the common language. This is done by the second network. The interesting point in the whole process is that at the end, a closed loop of information exists. The information is encoded and decoded. The network that encodes the information has to be trained with the same data set with which the decoding network was trained. Other types of encoding/decoding/storing information, such as associative memory networks, were considered, but seemed inappropriate for the task since they lacked the ability to synthesize *original* messages in the synthetic robotic language.

In a sense by using this system of two networks the agent is able to verify its learning process since the two networks, to an external observer, are auto-associative. Moreover besides receiving feedback from the environment (while observing the location of food sources), the agent will be able in future implementations to receive feedback from other agents as well during the demonstration of its language skills stage.

5.4 Information Fusion

This module collects data like distance, direction and food type from the neural networks experts. It combines them, and forwards food-related information to the error estimators, which in turn compare it with the actual food source position and type.

6 Connectionist Architecture Training

The supporting mechanisms for the language skill acquisition are trained both off-line and on-line. The self-organizing networks assist in the recognition of the basic language components (shapes) and are trained off-line; their training stops as soon as the learner agent starts observing the teacher. On the other hand, the expert sub-agents are trained on-line, while they observe the teacher. This training leads to the ability to understand and generate the language.

6.1 Kohonen Networks Overview

The self-organizing topological maps were introduced by Kohonen [10] and use an unsupervised form of learning. In this type of networks the organization is not imposed by the programmer but is devised by the network itself and it is a function of the organization of the input data presented to the network. The network is composed of nodes that store multi-dimensional weight vectors. When a multidimensional vector is presented to the network, the node whose weight vector is the closest (using either Euclidean or Manhattan distance) to the presented vector is updated and it gets a bit closer to the presented vector. Usually some of the neighbor units get updated as well in order to create node areas with similar weight vectors.

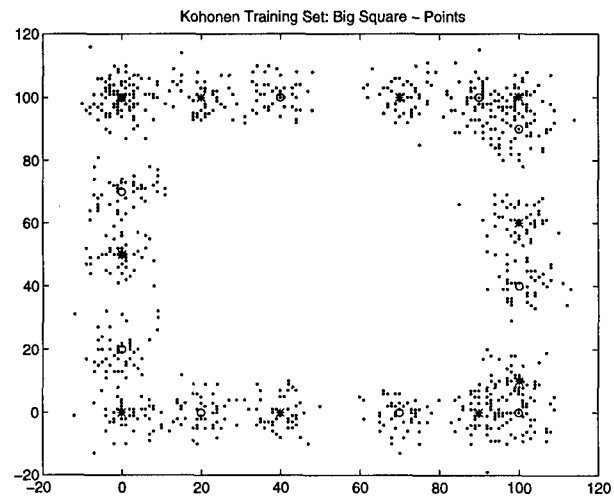


Figure 3: Distribution of Kohonen network training vectors describing a large square

The learning is unsupervised and during its process, node areas attract similar vectors, so areas can be labeled according to the vector type they attract. This type of networks is quite popular among roboticists because it provides an easy way of autonomous agent learning without feedback. Nehmzow, Hallam and Smithers [22] in-

troduced a project with Really Useful Robots that were capable of recognizing landmarks using self-organizing maps when presented with current sensor state and previous actions.

6.2 *Kohonen Network Training*

The training procedure of the Kohonen self-organizing networks was the following:

A simple modeler of the vehicles' characteristics was built that included their speed at 0.3 m/sec and turning velocities at approximately 2.8 rad/sec. This modeler simulated the shapes that the vehicle inscribe on the floor. The four distinct shapes were: large and small square (distance), triangle (direction), rhombus (food).

Ten points on the edge of the demonstrated shapes were chosen, each one of them forming a 20D vector, and the process was repeated 50 times for each shape but with up to 25% Gaussian noise added to the "ideal" shape, resulting in 200 training vectors.

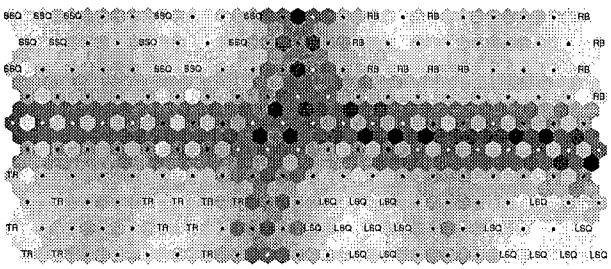


Figure 4: Kohonen Topographic Map: This is the actual map after training. The lighter the shade the more vectors are attracted in that note. The four different shapes regions are clearly seperated. The labels stand for: Large Square (LSQ), Small Square (SSQ), Triangle (TR) and Rhombus (RB).

Actual data of the vehicles' "dancing" were collected using the Visual Tracker and processed. Since it was not possible to collect a large amount of data using actual vehicles, five runs of each shape were made, and based on that 800 training vectors were produced (200 per shape). Again, Gaussian noise was added to the observed vectors of each shape. The output of the processed data for a large square is shown in Figure 3. Those 1000 20-dimensional vectors were used to train the Kohonen Network. Two training runs were performed, the first one with one pass through the data and the other one with ten passes. The first pass helped to create a uniform initial distribution of all the vectors in the Kohonen space. The second pass performed the actual training of the system.

Several different configurations of the Kohonen network were tested and the optimal topology for the given task was a 20 by 10 network, with 20D vectors in each node (one consideration when designing the network was the speed of the robots' processors, which are relatively slow compared to that of the workstations, where the initial simulations were run.) The units were arranged in hexagonal patterns and used a Gaussian distribution of neighborhoods. Sammon approximation of 20D to 2D gives a visualization of the shape vectors in space, in Figure 5.

As Figures 4 and 5 show, the Kohonen Self-Organizing Map gave very good results. The networks after training reached an organization that separates quite nicely the different shapes. In this way the system can easily distinguish between different types of shapes with a performance over 98%.

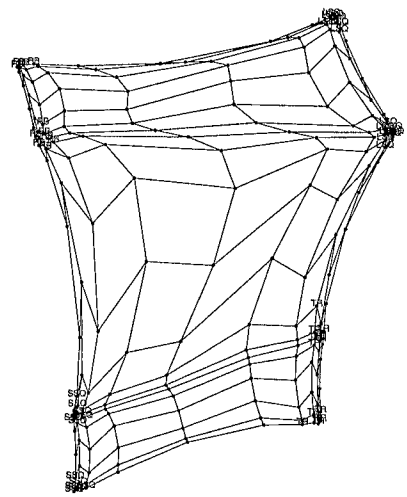


Figure 5: Sammon approximation of 20D to 2D vectors. Provides reasonable visualization of the clustering of the vectors in the space. The vectors in the four corners attract the different shapes.

6.3 *Back-propagation Network Design and Training*

The topology of all back-propagation networks was either one or two input layers, four hidden layers and either one or two output layers depending on the function of the specific expert.

The design of the back-propagation networks was implemented having in mind the necessity of training the networks on-line, on-board the robots. This requirement added a lot of overhead and effort to the design of the networks since the processor on the robots did not support floating point arithmetic.

Several different sigmoid functions were tested and the following gave the best results:

Actual Value	*Expected Value*	*Error (%)*
0.37	0.33	12.12%
0.71	0.67	5.97%
1.11	1.00	11.00%
1.23	1.33	7.51%
1.63	1.67	2.39%
1.96	2.00	2.00%
2.25	2.33	3.43%
2.55	2.67	4.49%
3.03	3.00	1.00%
3.25	3.33	2.40%
3.69	3.67	0.54%
4.11	4.00	2.75%
4.32	4.33	0.23%
4.73	4.67	1.28%

Table 2: Actual, Expected Values (converted into meters) and Error (%) in the distance estimation expert.

$$g(x) = \frac{1000}{1+e^{\frac{-x}{1000}}}$$

$$g'(x) = \frac{e^{\frac{-x}{1000}}}{(1+e^{\frac{-x}{1000}})^2}$$

For the training of the network a standard back-propagation algorithm was implemented. The algorithm compared the expected with the actual output, computed the error and trained the network by altering the weights between its nodes depending on how they affected the error.

The number of shapes for each language component was used as input, the food source positions or type as expected output and its difference as error (for instance in the distance network the number of large and small squares was used as input and the actual distance of the food source as expected output. The difference of the estimated from the actual food distance constituted the error). The same data were used the other way around for training the paired network (input the distance and output the number of shapes).

The data shown in Table 2 and Table 3 give the performance of the network after being trained with just eight pairs of shapes, food distance pairs. The ability of the network to perform well using this small amount of examples is crucial to the on-line and real-time nature of the experiment, since the language can be learned with just eight examples. The learning rate used was $\eta = 0.18$, the training set size was 8 and the repetitions over the set (epochs) around 600.

Figure 6 shows the mean error of all the acquisition experts plotted against the mean error of all demonstration experts. The acquisition experts began with a greater error than the demonstration experts (44% vs 27%). However the former converged much faster than the latter, having their error drop below 10% in just 100 epochs. The demonstration experts needed more than 450 epochs to reach comparable performance. The network reaches its optimal performance after 600 iterations.

Network: Skill	*Mean*	*Median*	*St. Dev.*
Acquisition	4.08%	2.57%	3.77%
Demonstration	8.17%	8.72%	3.65%

Table 3: Comparison of network performance. The acquisition networks have smaller error than the demonstration networks.

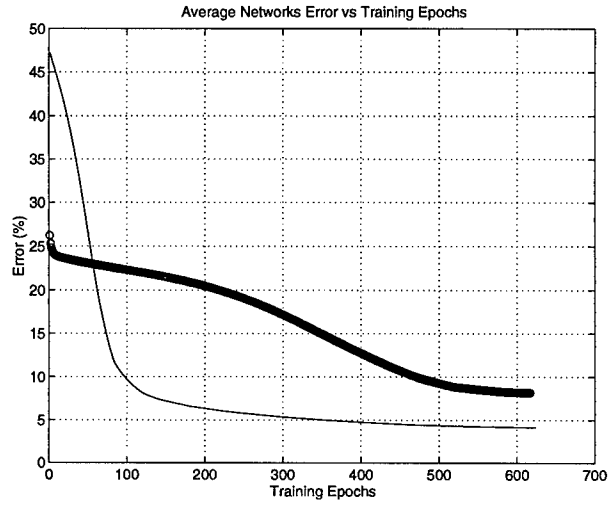


Figure 6: Diagram of Error vs Iterations. The thin line represents the performance of the acquisition skill networks and the thick line the performance of the demonstration skill networks.

Figure 7 and Table 2 show that the experts are very good at generalizing their knowledge. The results shown represent a validation experiment that included 14 different food locations. The obtained results are very close to the ideal solutions, proving the ability of the experts to generalize on the acquired language.

7 System Hardware

The main components of the experimental setting are a visual tracking system and a group of mobile robots. The visual tracking system uses a ceiling camera for tracking the positions of the robots, which carry LEDs (light-emitting diodes) on top, and then transmits information to them via a serial line.

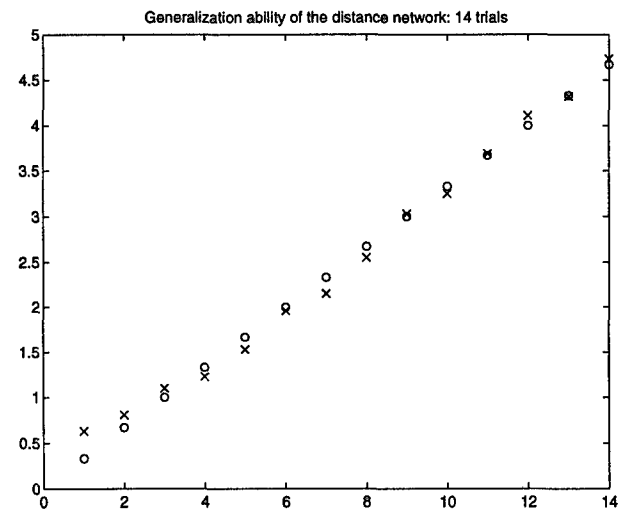


Figure 7: Diagram of expected food source positions distance (marked by o) vs actual food source positions distance (marked by x).

7.1 *Parallel Multiagent Visual Robot Tracking System*

Images are acquired by a monochrome CCD camera mounted on the ceiling above the experimental arena. The ceiling camera passes the video images to a specialized Transputer Board with three T800 transputer processors. The first one acts as the frame-grabber (Frame chip), as well as a server for a video monitor display; the second one, a graphics chip, controls an external Multi-Sync graphics display which shows the positions of the robots and the locations of the LED-tracking windows on top of them. Finally the third processor, the Root chip, is responsible for the coordination of the system and for running the main tracking algorithms. An overview of the tracking system can be found in Figure 8.

The tracking software was build upon an earlier system developed by Elliot [6] at the University of Edinburgh. It is enhanced with a multi-agent tracking capability, able to keep track of the positions of more than ten robots with an output frequency of at least 2Hz, and with a data logging system for later off-line processing. It consists of two parts, a scanner and a tracker.

The scanner fetches new images from the frame-grabber and searches them to find the robots in the image. As soon as the scanner finds an LED it assigns a "window" to it and starts searching around it at a fixed radius (the size of the robot) to find the other LEDs. Since the robots have two lights at the rear and one at the front the system is able to infer the direction of the robot. The window information is subsequently passed to the tracker which assigns a sub-process to each robot;

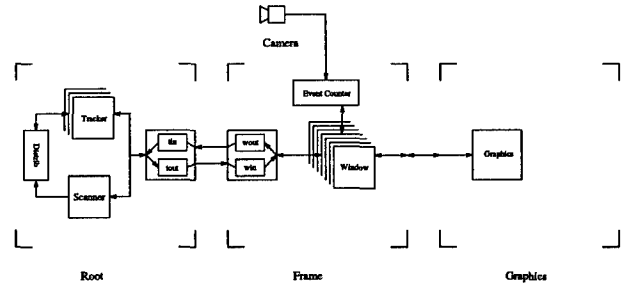


Figure 8: Overview of the Visual Tracking System

this subprocess "follows" the robot around as it moves.

7.2 *Mobile Robots*

The robots are designed using the LEGO Technic system as basis, along with specialized electronics designed at the the University of Edinburgh [5].

7.2.1 *Robot Design*

The LEGO robots are quite small, comfortably fitting inside a cube with 35 cm sides. Only two wheels are used in the design and are placed near the center of each robot with the battery near their axis. A third wheel is used to provide static stability to the robot. The symmetry of the robot design and the proximity of the wheels' axis to its physical center and to its center of mass enables the robots to be holonomic, to have one degree of freedom more than what is needed to explore their environment, or, in other words, to be able to turn without changing position, simply by rotating.

7.2.2 *Control Modules and Electronics*

The electronic hardware includes a processing board, known as the brain-brick, which is equipped with a commercial 68070-based processor, real-time clock, control for up to four stepper motors, a hall-effect sensor, six binary switches for bumper use and an I^2C bus for connecting various add-on boards.

The boards currently available include a Light Dependent Resistor (LDR) board with four photo-resistors (useful for behaviors like phototaxis or finding lines on the floor) that operate on a logarithmic scale and an infra-red board with infrared sensors (linear scale). The infrared sensors are capable of detecting obstacles at up to 1.2m distance and their detecting angle is around 25 degrees. Currently these modules communicate at a speed of 9.6kbps and can pass positioning information and various commands to each robot via a serial line or transceivers.

7.2.3 Programming the Robots

The robots' microprocessors can be programmed using a augmented multitasking version of C. All the programming was done using GNU's `gcc` cross-compiler. The source code was compiled on a SunOS4.x Sparc machine for the robot processor and then downloaded to the brain-brick via a serial link. One drawback of the brain-brick is that it does not have a floating-point co-processor and all arithmetic has to be converted to integer.

8 Experimental Domain

In this section we describe the experimental domain within which we demonstrate the learning process. Recall that the general sequence of events is as follows: One agent goes to search for a food source. When it finds one it returns to the home region and performs a dance that describes the position of the source, in terms of distance and angle relative to the home region and the type/quantity of the food. The learner agent observes the dance and tries to understand it. When the teacher agent goes out to reach the food source the learner agent follows, and tries to correlate the food source position and its type to the dance it observed previously. After a number of trials, the learner agent's predictions become sufficiently accurate for it to try to demonstrate its new ability by looking for food itself and then returning to the home region and dancing to demonstrate to the other agents the food source location.

8.1 Typical Experimental Cycle

We now describe a typical experimental cycle on our instantiation of the architecture.

1. The teacher robot performs a dance indicating food at a particular location and of a particular type. It is observed by the learner via a camera mounted above the teacher. The tracker searches for corresponding pairs of LEDs (one at the front and two at the rear of the robots). If it finds one it assigns to each light an active window which is then responsible for tracking its light continuously.

2. For speed's sake each pair of windows estimates the next position of the robot based on the five previous positions.

3. A module inside the Significant Events Detector reads pairs of window positions to find the "center" of the robot and stores it into an array.

4. Another module inside the Significant Events Detector checks to see whether the robot has completed a shape and has indicated this by pausing for more than two seconds. If this is the case then it fetches ten (x,y) coordinate pairs at even intervals from around the track of the shape and feeds them into the Kohonen network. The data is scaled so as to make the shape recognition independent of the speed of the robot. The starting position of the shapes does not have to be fixed, since it is calculated each time a new shape is inscribed. A shape does not have to be "perfect" and finish exactly where it started (erroneous turns, floor anomalies always exist in the real world).

5. The Kohonen network distinguishes between shapes as follows: first it finds the network node with the smallest euclidean distance from the input vector. If this node has a label (ie looks like a previously-seen shape) then it is selected. If not, then the network is presented with another vector from the same data set but with different starting position and possibly different intervals between data. If this does not work either the closest *labeled* vector is selected.

6. The shape counting daemons track the shape recognition process and store the number of shapes. The system loops until all shapes are completed, at which point there is a 10 second pause in the demonstrator's movements.

7. In the current instantiation, the demonstrator's visit to the food source is monitored by the tracker. This is more convenient for practical reasons, although the learner does possess the ability to follow the demonstrator. The position and type of the actual food source along with the number of repetitions of each shape are stored.

8. Because the experts are currently trained when all the data are collected, eight different food source locations are first found. This essentially corresponds to the learner remembering a series of teaching experiences before starting its learning process. Experiments were also conducted with the experts being trained incrementally, as they observed various examples, but this yielded diminished performance. When trained incrementally the agents decided when they were ready to demonstrate their knowledge of the language by evaluating how well they were predicting the positions of the food sources.

9. After the training the agent is able both to "understand" the language and to demonstrate it. If a random food source position is selected and presented to the experts, they are able to decide how many repetitions from each shape the agent should make. The learner agent using this information demonstrates the dance itself and goes to the food source.

9 Future Work

The language introduced here is synthetic, ie proposed by the authors. But the language could also be devised

by the interacting agents. Very briefly, this case involves agents that will evolve on-line and that will devise a language of their own (in a simulator). Agents that learn the language fast (ie the first pair that manage to communicate) will have a competitive advantage (food position) over the others that do not have such skills. This language dialect will carry greater momentum than the other dialects and should be able to establish itself. Similar work has been carried out by MacLennan [18] and Werner, Dyer [27].

Because the agents' world is inherently noisy and the agents' movements are limited by design factors a context dependent grammar might be proven useful. In state-based context dependency [31] (indexicality) the meaning of a word depends on the state of the world[1]. In this way an agent would need a smaller vocabulary of words to represent more concepts (internal representations). An agent can relate the context of a word either to the state of the world (sensory input) or to the adjacent words in the same phrase. A compositional language could be used in order to minimize the learning rate of new words. Compositional languages, by the use of grammar allow the meanings (internal agent representations) to be decomposed into their component words. In this way, the agents don't have to learn whole phrases each time, but can decompose the phrases into already known words and infer the meaning of single unknown words.

The architecture can be augmented in several ways to facilitate automatic generation of experts in the case of a more complex and/or evolving language. A system in which experts "vote" whether they can explain the elements of the language can be implemented, where successful experts will receive positive feedback, and unsuccessful negative. Based on the overall feedback the experts will be evaluated, having only the fittest survive. Another issue that can be addressed in the case of an evolving language is that of the Kohonen networks that recognize the actual elements of the language. In this implementation the networks are not trained on-line, and are able to recognize the off-line trained shapes only. An interesting experiment would be to continue the training of the networks during the actual experiments and train them to possible new shapes.

10 Summary and Conclusion

This paper puts forward a framework within which communication skills can be transferred from one agent to another and presents experiments demonstrating how a robotic agent can learn a simple movements-based language by observing a teacher. The learner agent is able to acquire the semantics of the language by associating the symbols with relations in the world and to generate new symbols which allow it to communicate with other robots.

Experimental performance is encouraging: the connectionist models distinguish accurately between different dance symbol types and learn quickly their associations with distances, bearings and locations measured in terms of the agents' own perceptions. Moreover the introduced architecture scales up easily by introducing new experts for coping with new language elements.

[1] For instance, in the phrase "look at this", the word "this" can have a variety of meanings [29]

References

[1] Dorothy Cheney and Robert Seyfarth. Animal communication and the study of cognition. In Carolyn Ristau, editor, *Truth and Deception in Animal Communication.* Lawrence Erlbaum Associates, 1991.

[2] Robert Cummins. *Meaning and mental representation.* MIT Press, 1989.

[3] Kerstin Dautenhahn. Getting to know each other – artificial social intelligence for autonomous robots. In *Robotics and Autonomous Systems 16*, pages 333–356, 1995.

[4] Gerald DeJong. An approach to learning from observation. In R.S. Michalski, J.G. Carbonnell, and T.M. Mitchell, editors, *Machine Learning: An Artificial Intelligence Approach: Vol II.* Morgan Kaufmann, Los Altos, 1986.

[5] J. Donnett and T. Smithers. Lego vehicles: A technology fo studying intelligent systems. In Meyer and Wilson, editors, *From Animals to Animats.* MIT Press, 1990.

[6] John Elliot. A framework for controlling toy cars. BS Thesis, Department of AI, University of Edinburgh, 1994.

[7] Bennett Galef, Jr. Imitation in animals: history, definition and interpretation of data from the psychological laboratory. In Zentall et al, editor, *Social Learning: Psycological and Sociological Perspectives.* Lawrence Erlbaum Associates, 1988.

[8] James Gould. Honey bee cognition. In C.R. Gallistel, editor, *Animal Cognition.* MIT press, Cambridge, Mass., 1992.

[9] Gillian Hayes and John Demiris. A robot controller using learning by imitation. In *Proceedings of the IRS-94, Grenoble, France*, 1994.

[10] T. Kohonen. *Self-Organization and Associative Memory.* Springer-Verlag, Berlin, 1989.

[11] Yasuo Kuniyoshi, Hirochika Inoue, and Masayuki Inaba. Design and implementation of a system that generates assembly programs from visual recognition of human action sequences. In *In Proceeding of the IEEE international workshop on Intelligent robots and Systems*, 1990.

[12] Joseph Levine and Kenneth Miller. *Biology, Discovering Life.* Heath and Company, New York, 1991.

[13] Bruce MacLennan and Gordon Berghardt. Synthetic ethology and the evolution of cooperative communication. *Adaptive Behavior*, 2:161–187, 1993.

[14] Maja Mataric. Designing emergent behaviors: From local interactions to collective intelligence. In *From Animals to Animats II.* MIT Press, 1993.

[15] Maja Mataric. Learning to behave socially. In *From Animals to Animats III.* MIT Press, 1994.

[16] Maja Mataric. Reward functions for accelerated learning. In *Machine Learning.* Morgan Kaufmann Publishers, 1994.

[17] David McFarland. *Animal Behaviour.* Longman, 1993.

[18] Bruce McLennan. Synthetic ethology: An approach to the study of communication. In Chris Langton et al., editor, *Artificial Life II, Proceedings of the Second Workshop on Artificial Life*, pages 631–658, 1990.

[19] Nicolas Meuleau. Simulating coevolution with mimetism. In Francisco Varela and Paul Bourgine, editors, *Towards a Practice of autonomous systems: Proceedings of the First European Conference on Artifical Life.* MIT Press, 1992.

[20] Robert Mitchell. A comparative-developmental approach to understanding imitation. In *Perspectives in Ethology vol. 7*, pages 183–215. 1987.

[21] Alexandros Moukas. Communication skills acquisition in interacting agents. In *Proceedings of the Easter School on Computational Modeling in Synthetic Psychology, Learning and Memory*, University of Cambridge, England, 1995.

[22] Ulrich Nehmzow, John Hallam, and Tim Smithers. Really useful robots. In *Proceedings of Intelligent Autonomus Systems 2*, volume 2, Amsterdam, 1989.

[23] T.D. Seeley. The honey-bee colony as a superorganism. *American Scientist*, pages 546–553, 1989.

[24] John Smith. Animal communication and the study of cognition. In Carolyn Ristau, editor, *Cognitive Ethology: The Minds of Other Animals.* Lawrence Erlbaum Associates, 1991.

[25] Karl von Frisch. *The Dance Language and Orientation of Bees.* Oxford University Press, 1957.

[26] Gerhard Weiss. Action selection and learning in multi-agent environments. In Jean-Arcady Meyer et al, editor, *Proceedings of the Second International Conference on Simulation of Adaptive Behavior*, pages 502–510, 1993.

[27] Gregory Werner and Michael Dyer. Evolution of communication in artificial organisms. In Chris Langton et al., editor, *Artificial Life II, Proceedings of the Second Workshop on Artificial Life*, pages 659–688, 1990.

[28] Mark Winston. *The Biology of the Honey Bee.* Harvard University Press, 1987.

[29] Holly Yanco. Robot communication. Master's thesis, Department of EECS, Massachusetts Institute of Technology, Cambridge, MA, USA, 1994.

[30] Holly Yanco and Lynn Andrea Stein. An adaptive communication protocol for cooperating mobile robots. In Stewart W. Wilson Jean-Arcady Meyer, Herbert L. Roitblat, editor, *From Animals to Animats II*, pages 478–485, 1993.

[31] George Yule. *The study of language.* Cambridge University Press, 1985.

The Evolution of Communication Schemes Over Continuous Channels

Gregory M. Saunders
The Institute for the Learning Sciences
1890 Maple Street
Northwestern University
Evanston, IL 60201
saunders@ils.nwu.edu

Jordan B. Pollack
Computer Science Department
Volen Center for Complex Systems
Brandeis University
Waltham, MA 02254
pollack@cs.brandeis.edu

Abstract

Many problems impede the design of multi-agent systems, not the least of which is the passing of information between agents. While others hand implement communication routes and semantics, we explore a method by which communication can *evolve*. In the experiments described here, we model agents as connectionist networks. We supply each agent with a number of communications channels implemented by the addition of both input and output units for each channel. The output units initiate environmental signals whose amplitude decay over distance and are perturbed by environmental noise. An agent does not receive input from other individuals, rather the agent's input reflects the summation of all other agents' output signals along that channel. Because we use real-valued activations, the agents communicate using real-valued vectors. Under our evolutionary program, GNARL, the agents coevolve a communication scheme over continuous channels which conveys task-specific information.

1. INTRODUCTION

Animals, both real and artificial must constantly interact with others by competing for limited resources, by cooperating on a difficult task, or by communicating information about the environment. This paper focuses on communication; in particular, on the issue of how a set of agents can *evolve* a communication scheme to solve a given task without a priori native structure in place.

In speaking about communication schemes, we wish to avoid the term "language." A communication scheme describes the actual signals passed between agents. Language, a collection of sentences drawn from a finite vocabulary, denotes an *interpretation* of a communication scheme. In this sense, language is a subjective phenomena and is ascribed by an observer (Kolen and Pollack, 1995). For instance, the description "agent 0 passes a value of 0.341 to agent 2 down channel 1" reflects the implementation of interaction, while the alternative description "agent 0 just signalled the presence of food to agent 1" involves the interpretation of interaction. These two terms, communication scheme and language, reflect different aspects of the same phenomenon. Their use depends upon which aspect of interaction – implementational or interpretational – we wish to emphasize.

The role of communication in multi-agent systems remains one of the most important open issues in multi-agent system design (Brooks, 1991; Arkin and Hobbs, 1993). Evolution, or genetic search, lends insight into these issues in two ways. First, as a practical matter, evolution opens the door to task-specific languages. Communication within a group of agents, robotic or simulated, should possess some level of flexibility. We resist the urge to adopt a single, fixed scheme for all tasks or to implement a new scheme for every task merely because we can easily understand these approaches. We have turned to evolution as our designer because it allows the possibility for communication schemes to emerge from the communicative needs of the agents actually solving a given problem with little or no regard for explanatory clarity.

The second motivation for studying the evolution of communication focuses on the means rather than the end. Understanding how groups of agents evolve a common communication scheme – knowing which aspects of internal state they choose to communicate, or how they represent information as messages, for instance – should provide useful insights into language development. Furthermore, such understanding would have practical consequences. If we find that our agents always adopt a similar communication scheme, then we should examine that language in detail. Perhaps it provides increased efficiency for the task, or is particularly robust. On the other hand, such

regularities could emerge from learning bias in the evolutionary algorithm.

2. RELATED WORK

Several other researchers have studied the evolution of communication schemes, but their work all shares an emphasis on *discrete* communication. Yanco and Stein (1993) investigate a simple "follow-the-leader" task in which one agent, the leader, receives a command which must be followed by a group of agents. The leader chooses one of n symbols to represent the command, broadcasts the symbol to the other agents, and the subordinates respond. A reinforcement algorithm governs both the encodings of the leader and responses of the subordinates; over time, a consensus emerges between the two.

Werner and Dyer (1992) describe a more complex environment in which simulated animals must communicate to find mates. Females, while stationary, can sense potential mates within a limited range and "call out" to them by emitting a signal. Males, wandering around the environment, lack the capacity to produce signals or see the females directly, but they can sense the females' signals and respond by moving toward them. Using a neural network representation for agents and a genetic algorithm for search, Werner and Dyer show that the sexes can agree on a common language.[1]

MacLennan (1992) adopts a higher-level view of language by defining an abstract task in which a group of agents must learn to communicate. Each agent possesses local information in terms of one of n symbols; it chooses a second symbol (from a set of n) to convey that information, and other agents must respond appropriately. Using finite state machines to represent agents and a genetic algorithm, MacLennan shows how the group of agents evolve a common symbol-symbol mapping.

Collins and Jefferson (1991, 1992) study AntFarm, a simulated ant colony in which agents must learn to communicate the presence of food. At each time step, an agent drops between 0 and 64 units of pheromone, which then diffuses throughout the environment as a signal to other ants. Although they have yet to evolve cooperative foraging, the work sheds some light on representational issues, in particular, on the use of neural networks as an agent representation.

Ackley and Littman(1994) is the closest in spirit to our work, though significantly more complex in its construction, and focusing mainly on issues of distributed evolutionary computation. The agents in their model operated on tracks and used discrete bit communication in 6 channels.

As stated earlier, all of this work focuses on discrete communication signals, with ensuing finite-sized[2] languages (2-20 for Yanco and Stein; 4-8 for Werner and Dyer; 8 for MacLennan; 65 for Collins and Jefferson and 6 for Ackley & Littman). Implicitly, all of these studies assume that each agent possesses a perceptual system capable of discriminating external events into discrete categories and that the "true" behavior control lies hidden behind such systems. Furthermore, some studies make an architectural distinction between the agent sending the message and the recipient (Yanco and Stein, 1993; Werner and Dyer, 1992; and to some extend MacLennan, 1992, in the sense that at any given time, there is a privileged agent attempting to convey its local information to the others).

3. COMMUNICATION WITH CONTINUOUS SYMBOLS

Our approach to understanding multi-agent communication differs from the work described above. Based upon our work in understanding where complexity arises in observations of dynamical systems, we believe that granting a discrete symbol system privileged position as a substrate for evolution is a confusion of levels (Saunders, Kolen, and Pollack, 1994). Consequently, rather than assume the transmission of discrete signals between agents, we provide our agents with continuous channels capable of supporting a variety of communication schemes. Furthermore, we make no architectural distinctions between transmitter and receiver.

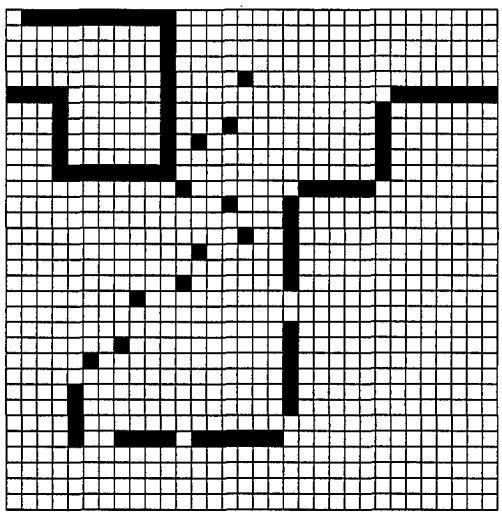
This section describes our main experiments which are fully described in Saunders(1994). First we briefly describe GNARL, the algorithm we use to evolve our agents. Then we introduce an extension of the Tracker task (Jefferson et al., 1992), which will serve as a substrate for our experiments. Next, we describe the method of communication our agents employ. Finally, we describe our experimental results.

3.1 GNARL

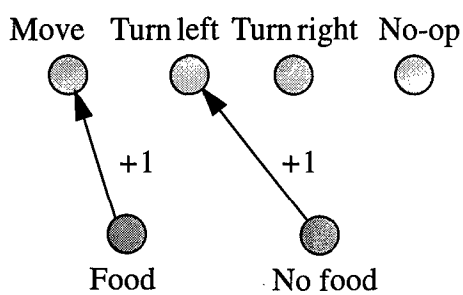
GNARL (Saunders, Angeline, and Pollack, 1994) is an algorithm based on evolutionary programming (Fogel, 1992) that induces recurrent neural networks. It pro-

1. Werner and Dyer (1993) propose a very interesting model "BioLand" which supports the evolution of communication as well, but the results focus on herding behavior rather than the evolved communication scheme, and it is unclear how the signals generated by the agents affect their behavior.

2. The size of a language is the number of distinct signals an agent may produce.



(a)



(b)

Figure 1: The Tracker task. (a) The trail is connected initially, but becomes progressively more difficult to follow. The underlying 2-d grid is toroidal; (b) The semantics of the I/O units for the ant network. The first input node denotes the presence of food in the square directly in front of the ant; the second denotes the absence of food in this same square. No-op, from Jefferson et al., allows the network to stay in one position while activation flows through recurrent links.

vides a mechanism for the simultaneous acquisition of network structure and weight values. GNARL employs a population of networks, replacing half each generation, and uses a fitness function's unsupervised feedback to modulate the amount of mutation applied to individual networks.

GNARL has been applied to several different problems (Angeline, Saunders, and Pollack, 1993). In particular, we have applied GNARL to the Tracker task (Jefferson et al., 1992) in which a simulated ant must learn to follow a broken trail of food (Figure 1a). Each ant receives two inputs: one indicating the presence of food in the square directly before the agent; and another detecting the absence of food in that same square. Jefferson, et al., allowed four primitive actions: move-forward (and implicitly eat food if present), turn left, turn right, and no-op (Figure 1b). Under these conditions GNARL evolved several different networks

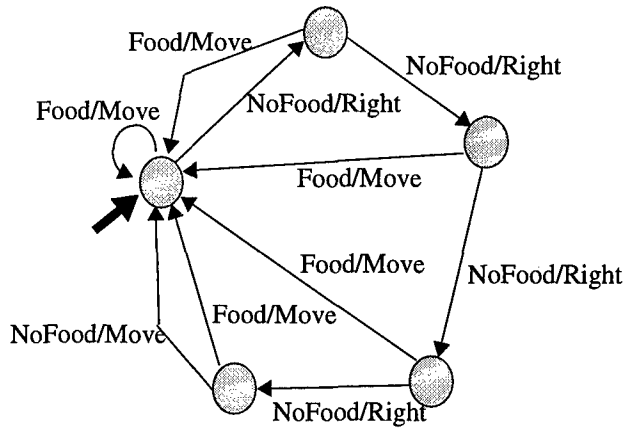


Figure 2: FSA hand-crafted for the Tracker task, from Jefferson, et al., 1992. The large arrow indicates the initial state. This simple system implements the strategy "move forward if there is food in front of you, otherwise turn right four times, looking for food. If food is found while turning, pursue it, otherwise, move forward one step and repeat."

as a solution, one of which closely approximates the finite-state automaton shown in Figure 2.[3]

3.2 *The Tracker Task, Revisited*

To study the evolution of communication in groups of agents, we extend the Tracker task in three ways:

- increasing the number of agents
- increasing the size of the grid to accommodate these agents
- moving all the food to a small area in the center of the environment

We assume that these modifications will shift the emphasis of the task from evolution of local internal state to evolution of distributed external state, i.e., communication. We concentrate the food within one area so that when an agent finds it and communicates, some food remains by the time other agents arrive. The size of the environment and the amount of food it contains far exceed the capabilities of a single ant: in the limited time available an ant can neither search the entire space nor consume all the food therein. Thus the task design ensures that the only method of complete success necessarily involves communication among the agents.

3.3 *An Architecture for Communication*

When faced with a task requiring communication, the architecture of Figure 1b will certainly fail; namely, be-

3. Note however that the network's behavior is not precisely captured by the FSA. Kolen (1994a, 1994b) shows that, in general, FSAs approximate networks only poorly. Another network induced by GNARL makes this point empirically. (See Saunders, Angeline, and Pollack, 1994).

cause it in no way supports communication. To remedy this shortcoming, we begin by adding n additional input and output units to the network of Figure 1b, representing n channels of communication. (These architectural changes, along with others not yet described, are shown in Figure 3.)

Output signals propagate throughout the environment, decaying in inverse proportion to squared distance.[4] Perception of these signals is governed by Equation 1. The input signal to agent a along the i^{th} channel, $s_{IN}(a, i)$, is a summation of the signals of all other agents along this channel. A is the set of agents, $s_{out}(b, i)$ is the i^{th} output signal of agent b. The noise in the channel, $U[-u_i, u_i]$ is a uniform random number with range specific to the channel, and σ is a linear threshold function, which bounds the signals in all channels to a user-specified range $[s_{min}, s_{max}]$. In the experiments below, $s_{min} = 0$ and $s_{max} = 40$.

$$s_{IN}(a, i) = \sum_{\substack{b \in A \\ b \neq a}} \frac{\sigma\left(s_{OUT}(b, i) + U[-u_i, u_i]\right)}{distance^2(a, b)}$$

Eqn 1

We have already demonstrated that when hidden nodes are added to the base architecture of Figure 1b, the resulting network can display complex behavior despite the simplicity of its move/turn outputs (Saunders, Angeline, and Pollack, 1994). In this study, however, we wish to maintain a clear separation between complexity arising from communication, and complexity arising from clever activation of the output nodes. We accomplish this in two steps. First, we condense the "move," "turn," and "no-op" outputs of the Tracker task into a single output unit: "Follow FSA." Second, we add n additional output units, representing the agents actions relative to the n communication channels. We maintain, from the original study, an implicit winner-take-all network on the (non-signal) outputs: when the "Follow FSA" node receives highest activation, the agent follows the primitive food-collection strategy of the FSA in Figure 2; when the i^{th} "Follow gradient" node receives highest activation, the agent follows the gradient of communication channel i. Figure 3 shows the final architecture. All activations are continuous; only the hidden activation is squashed (with the standard sigmoid function).

4. We assume that the signals propagate much faster than the agents react (as would a sound wave), so that effectively, at each discrete time step, an agent's output signals establish a wave front whose strength decays over distance.

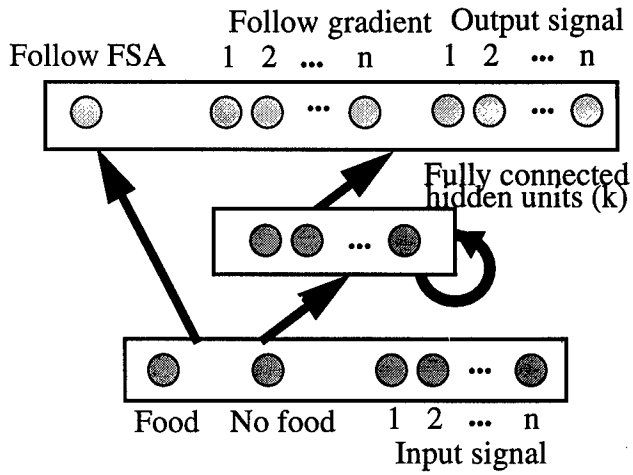


Figure 3: The semantics of the I/O units for evolving communication. The "food/nofood" inputs are from the Tracker task; the "Follow FSA" node represents one particular strategy found by GNARL. The additional nodes, described in the text, give the agent the ability to perceive, generate, and follow signals.

These modifications, though not essential to our results, greatly facilitate their analysis. The food collection strategy of the FSA is indeed quite simple; if activated repeatedly on a grid containing no food, the agent traverses its environment, turning in circles, but never veering from a straight line. Thus if we observe an agent moving non-linearly in the absence of food, we can assert with confidence that the agent is following a communication signal. Furthermore, because of the implicit winner-take-all network, we can easily observe *which* communication signal the agent is pursuing by simply comparing activations across the output nodes.

For the studies reported in this paper, all agents in an environment are homogeneous in that they share not only the architecture of Figure 3, but also common weights. As shown below, however, their behaviors will be quite different depending upon each agent's perspective of its world.

4. RESULTS

With this experimental setup, our thesis can be restated more precisely as follows. Multi-agent systems may evolve task-specific communication schemes. In particular, given the modified Tracker task (Section 3.2), a set of agents instantiated as recurrent neural networks (Figure 3), and a method of signal propagation (Section 3.3), then an evolutionary algorithm (GNARL, Section 3.1) is capable of evolving a communication scheme which allows the agents to perform their task.

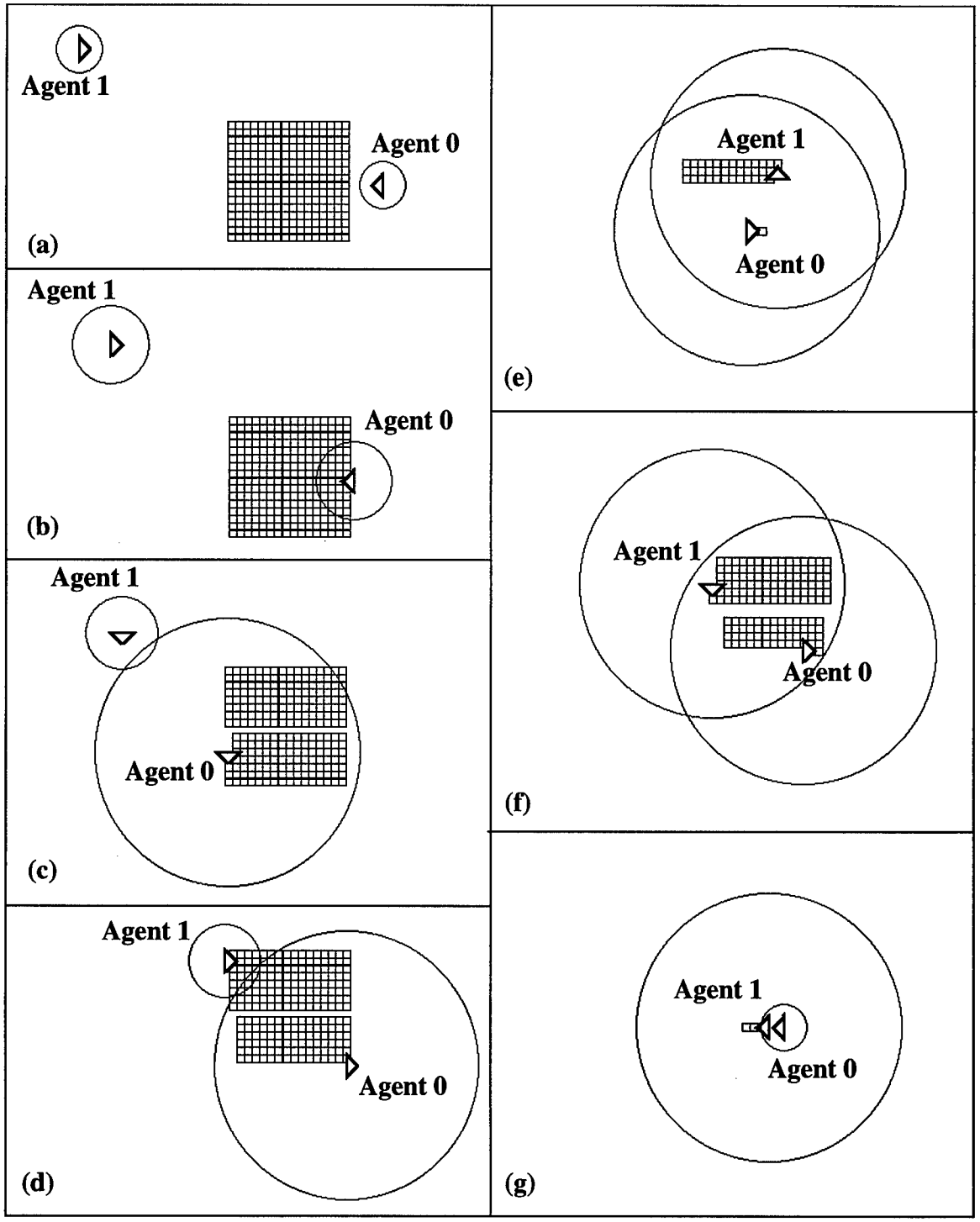


Figure 4: Scenes of evolved communication, 2 agents, 1 communication channel, no noise. The radius of the circles corresponds to the strength of communication. (a) Initial positions: neither agent can sense food; (b) One agent just reaches food, time is t=20; (c) Recruitment – first agent attracting the second, t=40; (d) Second agent just at food, t=60; (e) Agents eating, t=100; (f) Agents eating, t=14; (g) Recruitment – second agent attracting first, t=180.

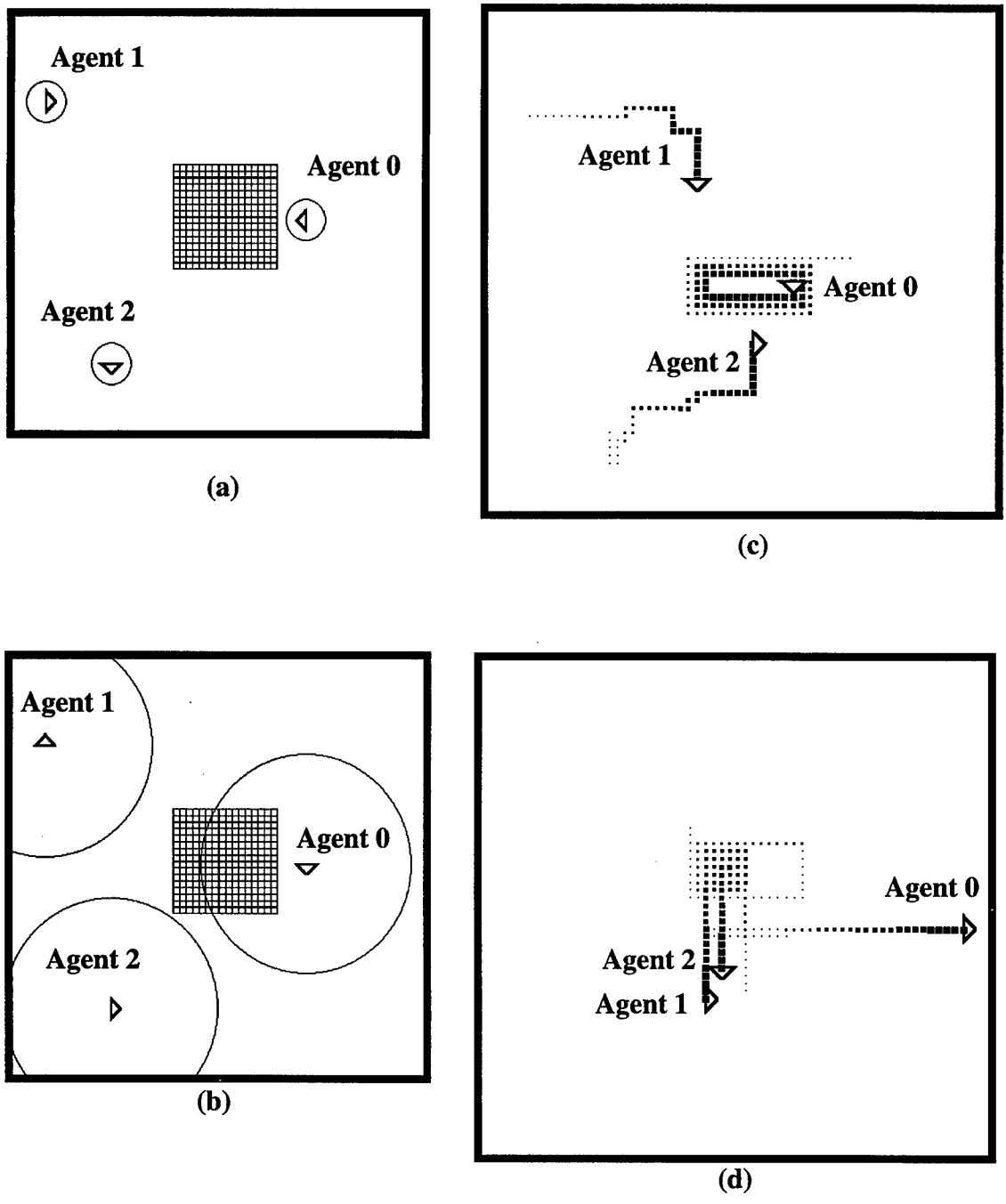


Figure 5: Scenes of evolved communication, 3 agents, 2 communication channels, one noisy. (a) Initial condition, circles denote signal 0; (b) After one time step, signal 0 has grown to maximum value. It oscillates between the two values when no food is present. (c) Recruitment, time 0 to 150. Dots indicate path of agents (food and signals have been removed for clarity); (d) Agent paths, time 150 to 300.

We begin testing this hypothesis with a very simple case: 2 agents, each with one hidden unit, capable of passing one real number between each other, with no noise ($u_0 = 0$). We measure fitness by simply observing the total amount of food eaten by the group. Figure 4a shows the initial environment. Without communication, each agent would follow the FSA, and agent 1 would move in a straight line, finding no food.

With communication, however, the story is quite different. In 800 generations, GNARL discovered a pair of agents (from a population of 50 pairs) which had learned to communicate the presence of food. Figure 4b shows the case just as agent 0 reaches the food; and Figure 4c shows recruitment: agent 0's strong signal, due to the food, attracts agent 1.[5] Figures 4e and f show both agents are emitting high signals while eating, and finally in Figure 4g, recruitment occurs again, this time in reverse.

We chose this case as a demonstration for several reasons. First, snapshots easily capture the evolved communication scheme: larger circles imply a higher signal. Second, the evolved language is fairly intuitive: each agent "yells" when it finds food by increasing the strength of its output signal; upon "hearing" such a signal, the second agent follows it to the source of food. We have also observed several different implementations of the same behavior, another common one being "Yell constantly when you're searching for food, but then grow quiet when eating." In this second case, agents learn to respond to silence.

We now focus in detail on a third, more complex communication scheme. For this experiment, we used the same food distribution, increased the number of agents to three, and retained a a single hidden unit for each agent. To investigate how the agents would respond to noise, we gave them two communication channels, the first clear (u_0=0), the second noisy (u_1=10). Figure 5a shows the initial environment. The circles reflect the strength of signal 0. We do not include signal 1, in the graphics, because it was not used by the agents (more on this below). After one time step, the signals along channel 0 have grown to their size in Figure 5b. In the absence of food, signals in this channel oscillate between these two extreme values. Figure 5c shows recruitment by agent 0; Figure 5d shows that recruitment is not permanent: when the food has been consumed, agent 0 strikes out on its own.

5. The circles denote not signal range, but the radius at which signal strength is one. (Signal strength is the summand in Equation 1.)

Figures 6-8 show how behavior is accomplished. Figure 6 gives the profile of agent 0 over the run. Note how its output signal 0 oscillates in the absence of food. Figure 7 shows the profile of agent 1 throughout the run. The lack of oscillation in agent 0's output is enough to turn agent 1 towards the food. (The 5 spikes in the behavioral profile indicate "Follow signal 0" behavior.)

Agent 2, however, is slightly different (Figure 8). Note the oscillation in its behavior, as it alternates between following the gradient of signal 0 and following the FSA. At first glance, this seems incorrect, because the inputs to agents 1 and 2 look identical, and their architectures are identical, but their output behaviors are very different. The problem might simply be one of perceptual scale (i.e. we can't see any differences). Figure 9 zooms in on the first 50 time steps of the signal 0 input to agents 1 and 2, and shows that there is a slight variance in magnitude of signals, and a phase reversal. Further sensitivity testing, by artificially varying the input signals and observing the resulting agent behavior, showed that the difference in behavior between agent 1 and 2 was caused by the phase difference, not the magnitude difference.

5. ANALYSIS

Detecting the presence of communication is more difficult than it sounds. Communication can occur across long and short distances of both space and time. Random noise can corrupt or masquerade as communication. To operationalize the effects of communication, we first adopted the following definition: *task-specific communication occurs between agents if performance drops when the communications channel is blocked.* In our experimental milieu, we blocked the agents' signals by shunting the channel with various constant values. In all cases, removal of channel 0 drastically reduced fitness, yet the removal of channel 1 failed to hamper the search behavior of the agents, confirming that the agents had learned to rely on the clear channel and ignored the noisy channel.

This definition of task-specific communication, however, makes two assumptions which limit its strength. First, it assumes that interagent communication is goal-directed; communication is the *means* to increasing the performance of a given task. Observing the behavior of the agents does not tell us why one agent squawks over a channel or why another agent reacts to the ruckus. Second, this definition assumes that an agent ignores channels providing it with irrelevant information, i.e., noise. Without dissecting the agent, one cannot tell if noise is a necessary environmental regularity contributing to the normal behavior of an agent. If this is true, blocking

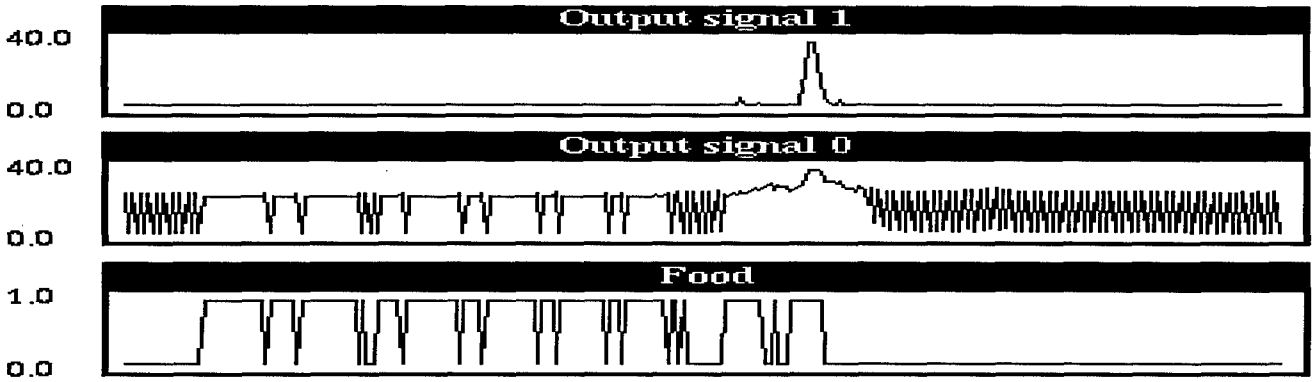


Figure 6: Profile of agent 0, for 300 time steps. The lowest graph is the food input: when food is detected, the value spikes to one; otherwise it is zero. This agent has learned to correlate oscillation of its output signal 0 with the presence of food.

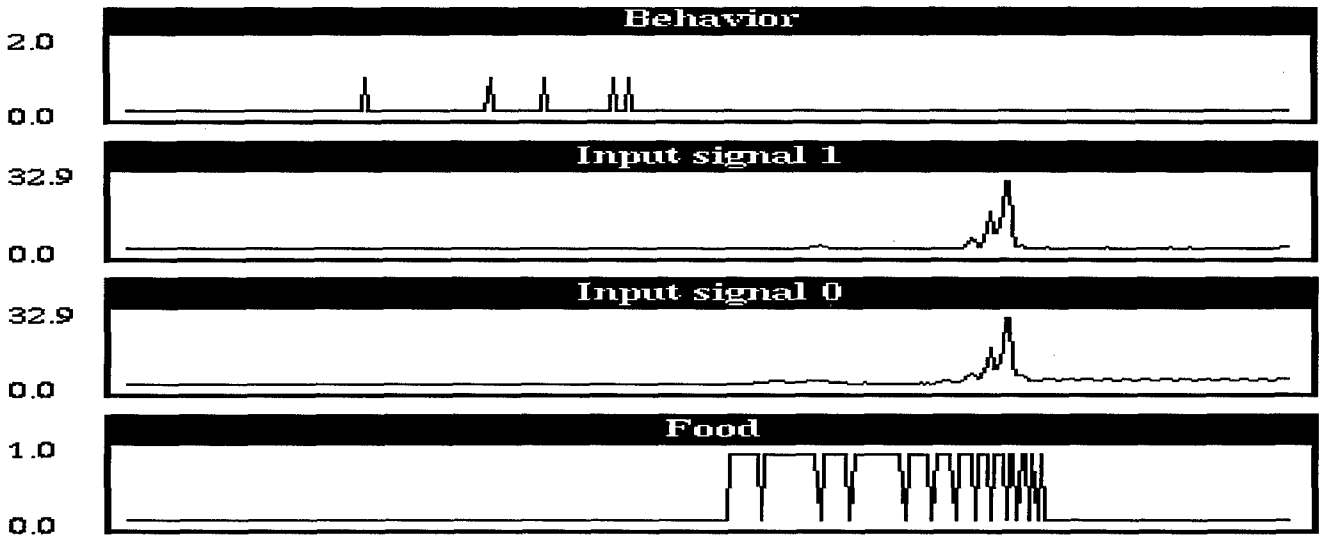


Figure 7: Profile of agent 1. The five spikes in behavior indicate points where the agent follows signal 0, as can be seen in Figure 5c. Because the agent perceives no food during this time, the resulting behavior occurs due to the agents input signals.

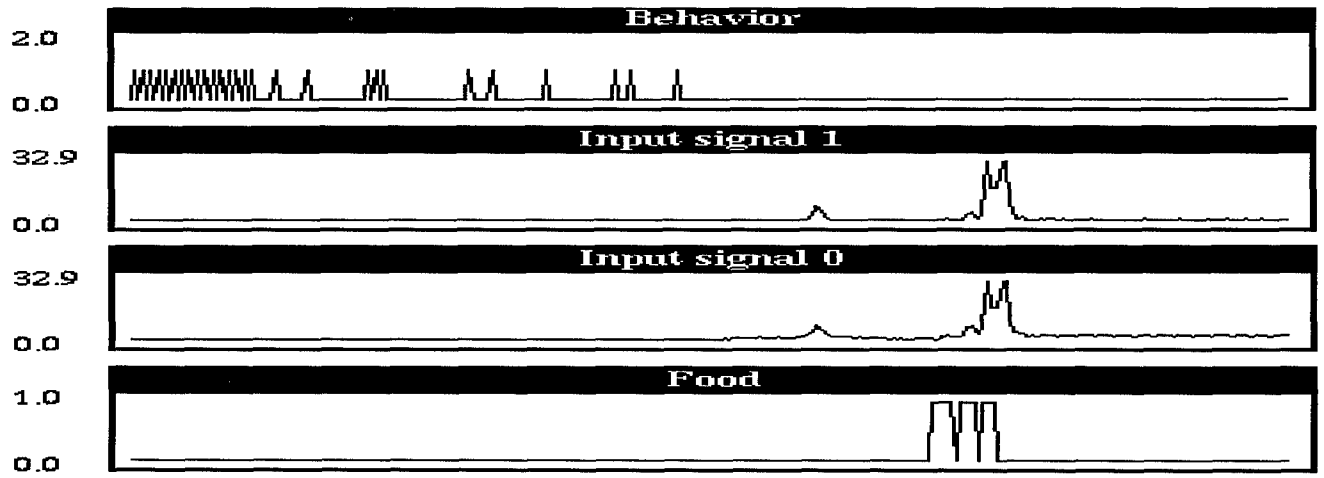


Figure 8: Profile of agent 2. Although its initial inputs (food & signals) look identical to that of agent 1, this agent's initial behavior oscillates between "Follow food" and "Follow signal." The difference is resolved in Figure 9.

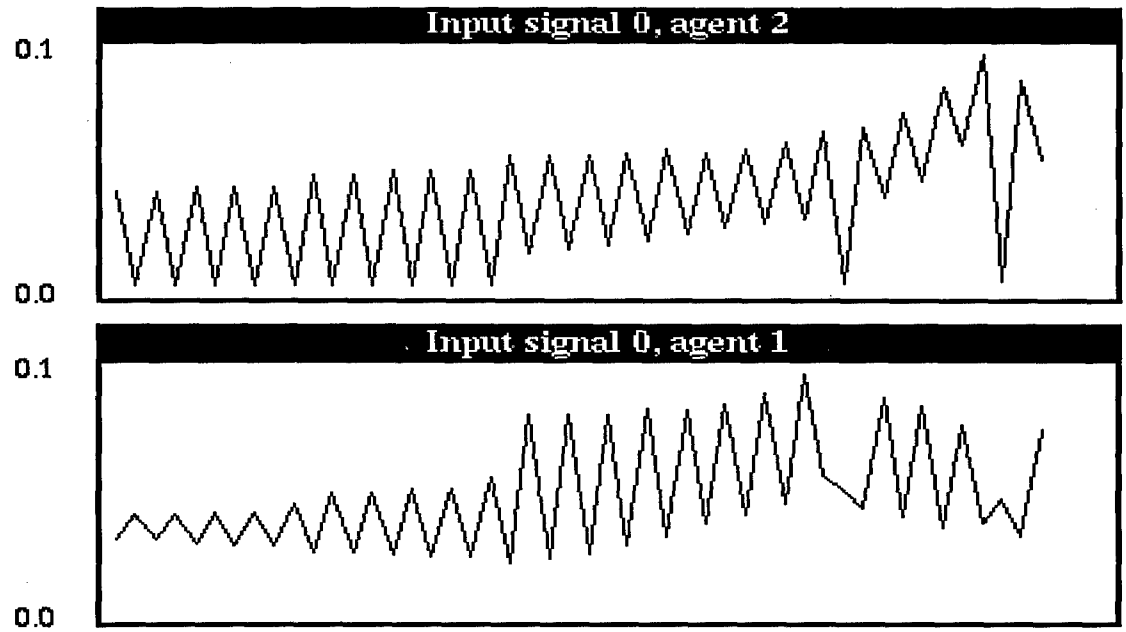


Figure 9: Close up view of the first input signal of agents 1 and 2, epochs 0 to 50, to see why their initial behaviors differ. Agent 1's input begins oscillating between .03 and .04. Agent 2's input begins oscillating between .06 and 0. Upon investigation, we discovered that it is not the magnitude, but the difference in phase which is responsible.

such a channel would more resemble severing an appendage rather than masking a sensor. In short, an evolved or programmed solution to a multi-agent problem is an *instance*, and drawing conclusions about the *class* from a single instance is risky business.

The switch from discrete to continuous signals brings into question the traditional notion of communication in this context. Recall the set of experiments involving three agents. An interesting communication scheme emerged which employed both constant and oscillatory signals. While one could claim that the agents learned to discriminate between oscillatory and constant signals or discern phase differences, we believe another mechanism is at work. Rather than recognize environmental patterns, the agent allows the input sequences to modulate its behavior-producing mechanisms. Kolen (1994b) used this approach to explain the behavior of recurrent neural networks. From this perspective, we view the agent as a state transform system consisting of a set of functions mapping internal state to internal state. Input, by this approach, selects the current transform from this set. At no time is the input stream partitioned, normalized, or recognized, it simply modulates the behavior of the network.

We began with very few assumptions about the nature of communication, essentially stripping away the information-theory veneer that has made previous systems easy to understand. First we replaced the engineer with evolutionary search. Second, we eliminated discrete events and allowed the agents to modify channels with continuous values. These assumptions did not prevent solutions to the modified Tracker problem, in fact some novel approaches were discovered. Identifying the contribution of communication to solving the task proved to be very difficult. Despite these difficulties with understanding how the agents operated, we were able to evolve agents which demonstrated such task-specific behaviors as recruitment. Our hope is that this work opens the door to the study of evolving continuous communication schemes.

Acknowledgments

This work was performed when both authors were with the Lab for AI Research at Ohio State University, along with Peter Angeline, John Kolen and Edward Large, all of whom greatly influenced the work. Partially supported by ONR grant N00014-92-J1195 to the second author.

References

Ackley, D. H and M. L. Littman (1994) Altruism in the evolution of communication. Proceedings of ALIFE IV. Cambridge: MIT Press 40-49.

Angeline, P., Saunders, G., Pollack, J. (1994). An evolutionary algorithm that constructs recurrent neural networks. *IEEE Transactions on Neural Networks*, 5(1):54-65.

Arkin, R. C. and Hobbs, J. D. (1993). Dimensions of communication and social organization in multi-

agent robotic systems. In Meyer, J. A., Roitblat, H. L., and Wilson, S. W., editors, *From Animals to Animats 2: Proceedings of the Second International Conference on Simulation of Adaptive Behavior*, pages 486–493, Cambridge. MIT Press.

Brooks, R. A. (1991). Challenges for complete creature architectures. In Meyer, J. A. and Wilson, S. W., editors, *From Animals to Animats: Proceedings of the First International Conference on Simulation of Adaptive Behavior*, pages 434–443. MIT Press.

Collins, R. J. and Jefferson, D. R. (1992). AntFarm: Towards simulated evolution. In Langton, C. G., Taylor, C., Farmer, J. D., and Rasmussen, S., editors, *Artificial Life II*, pages 579–601. Addison-Wesley.

Fogel, D. B. (1992). *Evolving Artificial Intelligence*. Ph.D. thesis, University of California, San Diego.

Jefferson, D., Collins, R., Cooper, C., Dyer, M., Flowers, M., Korf, R., Taylor, C., and Wang, A. (1992). Evolution as a theme in artificial life: The Genesys/Tracker system. In Langton, C. G., Taylor, C., Farmer, J. D., and Rasmussen, S., editors, *Artificial Life II: Proceedings of the Workshop on Artificial Life*, pages 549–577. Addison-Wesley.

Kolen, J. F. (1994a). Fool's Gold: Extracting finite state machines from recurrent neural networks. In J. D. Cowan, G. Tesauro, and J. Alspector, editors, *Advances in Neural Information Processing 6*. Morgan-Kaufmann.

Kolen, J. F. (1994b). Recurrent Networks: State Machines or Iterated Function Systems. In M. C. Mozer, P. Smolensky, D. S. Touretzky, J. L. Elman, and A. S. Weigend, editors, *Proceedings of the 1993 Connectionist Models Summer School*. pages 203-210. Hillsdale, NJ: Erlbaum Associates.

Kolen, J. F., and Pollack, J. B. (1995). The observers' paradox: apparent computational complexity in physical systems. *Journal of Experimental and Theoretical Artificial Intelligence*, 7, 253-277.

MacLennan, B. (1992). Synthetic ecology: An approach to the study of communication. In Langton, C. G., Taylor, C., Farmer, J. D., and Rasmussen, S., editors, *Artificial Life II*, pages 631–658. Addison-Wesley.

McDermott, D. (1981). Artificial intelligence meets natural stupidity. In Haugeland, J., editor, *Mind Design: Philosophy, Psychology, Artificial Intelligence*, chapter 5, pages 143-160. MIT Press, Cambridge, MA, reproduced from *SIGART Newsletter*, No. 57, April 1976.

Saunders, G. M., Angeline, P. J., and Pollack, J. B. (1994). Structural and behavioral evolution of recurrent networks. In J. D. Cowan, G. Tesauro, and J. Alspector, editors, *Advances in Neural Information Processing 6*. Morgan-Kaufmann.

Saunders, G. M., Kolen, J. F., and Pollack, J. B. (1994). The importance of leaky levels for behavior-based AI. *From Animals to Animats 3: Proceedings of the Third International Conference on Simulation of Adaptive Behavior.275-281*

Saunders, G. M. (1994) The Evolution of Communication in Adaptive Agents. Ph.D Dissertation Columbus: Computer & Information Science Department, Ohio State University. Available in ftp://nervous.cis.ohio-state.edu/pub/saunders/Dissertation/

Werner, G. M. and Dyer, M. G. (1992). Evolution of communication in artificial organisms. In Langton, C. G., Taylor, C., Farmer, J. D., and Rasmussen, S., editors, *Artificial Life II*, pages 659–687. Addison-Wesley.

Werner, G. M. and Dyer, M. G. (1993). Evolution of herding behavior in artificial animals. In Meyer, J. A., Roitblat, H. L., and Wilson, S. W., editors, *From Animals to Animats 2: Proceedings of the Second International Conference on Simulation of Adaptive Behavior*, pages 393–399, Cambridge. MIT Press.

Yanco, H. and Stein, L. A. (1993). An adaptive communication protocol for cooperating mobile robots. In Meyer, J. A., Roitblat, H. L., and Wilson, S. W., editors, *From Animals to Animats 2: Proceedings of the Second International Conference on Simulation of Adaptive Behavior*, pages 478–485, Cambridge. MIT Press.

Using A-Life to Study Bee Life: The Economics of Central Place Foraging

P. de Bourcier
Millennium Interactive Ltd., Quern House, Mill Court, Great Shelford, Cambridge CB2 5LD. U.K.
e-mail: peter@millennium.co.uk

Abstract

In this paper I model behavior exhibited by bees and ants when central place foraging. I evolve the choice of how many food sources the animat should visit in order to maximize either energy gain per time, or energetic efficiency. An accurate model of the energetic costs and time required to fly to the flower patch, collect nectar, and return to the hive results in a model that is more realistic than those created using more traditional techniques. The increased realism in this model allows me to highlight a discrepancy between proposed theoretical models of the economics of bee foraging and the observed behavior of bees.

1 Introduction

Potentially one of the most useful applications of A-Life, and Simulation of Adaptive Behavior techniques is in developing tools for biological inquiry. In this work I utilize simulation techniques to address the problem of diminishing returns for central place foragers. I describe a simulation based on behavior exhibited by various bees, wasps, and ants during foraging. Data on the mass, carrying capacity, and energy usage of bees are used in a model of the behavior of central place foragers finding, selecting, and feeding on flowers. Using this simulation I investigate the issue of when a central place foraging animat should return from a foraging trip. Bees often return from a foraging trip with less nectar than they could carry if fully loaded. If the bees were maximizing profit (net energy gain per time) they would always return with a full crop. It has therefore been postulated that bees attempt to maximize energetic efficiency when foraging. That is they maximize the gain of energy to the hive for the amount of energy they expend during the trip. In this simulation I evolve the decision of when an animat should stop foraging and return to the nest, under various circumstances, and compare the results from when the animat is attempting to maximize energetic efficiency to when it attempts to maximize net energy gain.

2 Biological Background

In social insects reproduction occurs through production of sexual forms (queens and drones) after the colony has grown in worker numbers. Development during phases of population growth is crucial to colony fitness as the size of worker force at maturity correlates positively with the number of reproductive forms that can be produced (Cole, 1984; Lee & Winston, 1987). However, most of the colony members (the workers) do not themselves reproduce but contribute to colony development by performing different duties. Young workers perform most of the duties inside the hive (feeding larvae, cleaning cells, and building combs) whereas older individuals leave the hive to forage. These foraging activities in turn provide the resources necessary to raise young and to maintain hive activities, thus allowing for colony growth and eventual reproduction. It would therefore be expected that the foraging behavior of workers would be highly developed, as it is under strong selective pressure.

Bees, wasps and ants have all evolved a repertoire of behavioral traits that allows them to locate food patches, forage on them and successfully return home. In the following sections I describe some of this behavior.

2.1 Dead Reckoning

One of the navigational skills available to hymenoptera[1] is most clearly visible in the desert ant (*Cataglyphis*) which has the ability to follow amazingly straight homeward courses which lead it over novel territory not covered during its circuitous outward journey (see figure 1). This novel route following implies that the ant does not have to rely on pheromone trails or visual landmarks as aids to relocate its home, but instead makes use of a dead reckoned home vector. Such a strategy could provide the animal with a continuously updated homeward bound vector, always connecting the animal with the starting point of its foray. When displaced

[1] Hymenoptera: The insect class that includes bees, wasps, and ants.

during its return trip the desert ant will run to a position displaced relative to the nest by the same vector as was used to displace the ant. Having reached the position which the dead reckoning system suggests is the nest, the desert ant starts performing a systematic search behavior that will enable the ant to eventually find the nest. Path integration and the use of vector information have been described for bees (von Frisch, 1967) as well as for other arthropods (Goerner & Zeppenfeld, 1980; Seyfarth *et al.*, 1982; Hoffman, 1983; Goerner & Claas, 1985; Mittelstaedt, 1985; Ugolini, 1987).

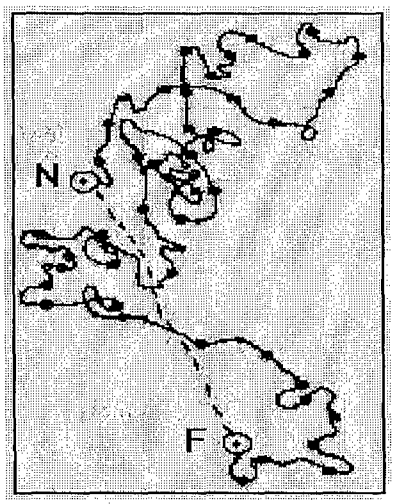


Figure 1: The foraging and return path of the ant, *Cataglyphis albicans*. The thin solid line marks the ant's outward path from its nest (N). The circle at the bottom (F) indicates the position of a food source. The ant returns to the nest along the path indicated by the dotted line. Based on Papi (1992, p.4).

The dead reckoning carried out by these insects cannot rely solely on integration of proprioceptive information. Small errors in the estimation of distance, and more importantly heading, would soon escalate. Insects have to use external stimuli to help keep these errors to a minimum. Celestial compasses may help to reduce the error in heading estimation and reduce the compounding effect these errors have. The bee's and ant's celestial compass exploits two sources of information; the direct light from the sun (Camhi, 1984. p.121) and the scattered light from the sky (Camhi, 1984. p.122). Both are used by the insect to determine a reference direction to which all rotary movements are related. This reference direction is the solar meridian, the perpendicular from the sun down to the horizon. The distance traveled is thought primarily to be recovered using rate of retinal image motion, though in ants the number of steps, and in bees the energy expended may possibly be used as well (for a review see Papi, 1992. p.99).

2.2 Flower Selection, Nectar Collection and Returning Home

When a bee reaches a forage patch where the density of flowers is sufficiently high for individual foragers not to be able to remember the positions of individual flowers, its movement patterns from flower to flower can be described by two parameters: the change in direction and the flight distance. The change in direction is the angular directional difference from one flower to the next in relation to the direction of approach. The flight distance is defined as the linear distance between successively visited flowers. Pyke (1980) describes the distribution of the change in direction and flight distance for *Bombus appositus* queens foraging on *Delphinium barbeyi* (larkspur). He suggests there is a considerable positive correlation between the directions of any two successive flights. He also suggests that there is a strong correlation between mean flight distance and mean nearest-neighbor distance between flowers. In other words, bees have a tendency to select flowers that are not far off their heading and are close. There are various complications to this state of affairs such as a varying dependence based on the quality of the recent rewards (for a review see Waddington & Heinrich, 1981).

Once on a flower, the bee collects the available nectar or pollen. An energetic model of foraging has been devised by Hodges (1981) and is described in a later section. The process of flower selection and nectar collection during foraging is repeated until the bee 'decides' to return to the nest to unload its crop. It is worth noting that bees often return to the hive with less than the maximum load they could carry. Krebs & Davies (1993, p.54) suggest that the bee experiences a curve of diminishing returns. They explain that this is because the weight of nectar in the crop adds an appreciable energetic cost to flight. Therefore the more the bee loads up its crop the more of its load will be burned up as fuel before it gets home. While foraging the gross quantity of nectar harvested increases at a constant rate, but the net yield of energy for the hive increases at a diminishing rate as the crop fills.

Schmid-Hempel *et al.* (1985) found that by increasing the distance to the flower patch or the inter-flower distances within the patch, he could influence the bee's decision about when to go home and empty its crop. When the distance to flowers and their inter-flower distance were increased bees were found to go home with smaller loads. He concluded that the bees' decision of when to unload had the effect of maximizing energetic efficiency (gain/loss) rather than net rate of energy increase (gain-loss) as was previously assumed. The experiments described in this paper address this issue.

2.3 Systematic Search

Once the insect has decided to return to the nest it attempts to minimize its dead reckoned vector. When the insect's home vector has been reset to zero the insect should have arrived back at the starting point of its journey. However due to navigational errors that accumulate during the entire round-trip, this is not necessarily the case. If this is true then the insect could engage in a random search for the nest, i.e., wander about like a particle in Brownian motion. Such random search, as discussed for migratory vertebrates, both fish (Saila & Shappy, 1963) and birds (Wilkinson, 1952), will be successful, given unlimited time. In real life however, there are inevitable time constraints imposed on the homing animal, so that a premium will be put on any type of systematic search by

which the probability of relocating the starting point can be increased beyond the random search level.

The geometry of such a search pattern has been studied comprehensively in the desert ant (*Cataglyphis fortis*) (Wehner & Srinivasan, 1981; Wehner & Wehner, 1986). The ant's search pattern consists of a number of loops of ever increasing size, starting and ending at the position where the home vector suggested the home site should be located. Thus the ants spend most of their time searching at locations where home is most likely to be. The search density rapidly decreases with increasing distance from the origin.

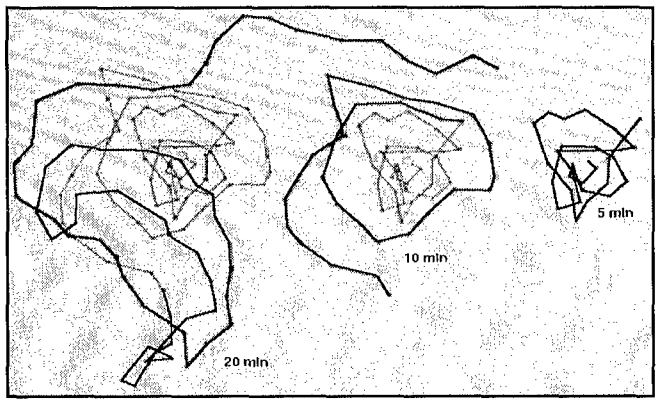


Figure 2: The search path of the desert ant, *Cataglyphis fortis*. The sequence of figures shows how the search pattern develops as search time proceeds. Based on Papi (1992, p.103).

The path taken by the ant is described as a 'cryptic spiral' (Wehner & Wehner, 1986) hidden behind a system of loops and turns (see figure 2). The ant's search pattern can be described thus:

- Search along a spiral path which occasionally changes its sense of rotation
- Interrupt this spiral path after a while and return to the center
- Once at the center spiral out again along a wider spiral path.

Purely egocentric systems for navigation such as dead reckoning are prone to cumulative errors. Resorting to systematic search is the animal's final emergency plan designed to cope with such errors.

3 Experimental Model

Using a simulation based on behavior described in the previous section I designed a central place foraging animat. In the following section I describe the implementation of this behavior, the energy model I used, and the task I set the animat.

3.1 The Task

The task I set for the animat was to collect food from a distant site. I set up simulated 'flowers' that contained a food source that I considered analogous to nectar. The flower site consisted of an area distant from the home position from which the animat started and hopefully finished a foraging excursion. The number of flowers at this site was set by the experimenter together with the area and position of the foraging patch. The flowers were randomly distributed within the foraging patch. Each flower had a uniform amount of nectar contained within it. The amount of nectar and its caloric value were also under experimental control.

At the beginning of each foraging trip the animat is given a vector that points to the center of the food cluster. The biological analogy to this vector would be either a vector learned from past experience or a vector communicated between individuals in the dance language of bees (von Frisch, 1967).

To find the food patch the animat must use a dead reckoning system. This system is implemented using both geocentric information and proprioceptive information. The animat has the ability to integrate the movements of its simulated wheels; this is done using integrated geometry. Wheel slippage is simulated in its simplest form by the addition of random noise to the wheel speed. As errors compound very quickly the animat has access to a compass. This constantly resets the estimated heading to the animat's true geocentric heading. This is analogous to use of a celestial compass. Using these mechanisms the animat constantly keeps a record of its approximate position in relation to the home position. Just as the insect can use this dead reckoned information on both the outward and homeward leg of its foraging journey so can the animat. The foraging patch can be found by attempting to minimize the difference between the forage patch vector and the inverse of the home vector.

Once at the foraging patch the animat must select a flower to forage from. Here the choice is made using the distance of the flowers in the immediate vicinity and their angle in relation to the animat's heading. This reflects the mechanisms postulated to be at work in the honeybee. Once the animat has 'decided' to return home to empty its crop it attempts to minimize the home vector that has been integrated during both the outward leg to the foraging patch and the numerous trips between flowers.

Once this vector has been minimized the animat will be in the rough vicinity of the home position. Small errors in wheel velocities cause the position of the animat to stray from the actual home position. The animat must then initiate a systematic search behavior in order to locate the home position. This behavior is modeled on that of the ant (Wehner, 1992). By attempting to keep its heading at just over 90° to the dead reckoned home vector the animat spirals out from its estimated home position. After a while the animat attempts to minimize the home vector again and returns to the position it thought the home was located at (though this will now be

shifted slightly due to errors in dead reckoning). Once there the animat may reverse direction (random choice with a probability P=0.5) and spiral out again with a greater angled spiral for a longer duration. This behavior is repeated until the animat finds the real home position.

Once at the home position the crop is notionally unloaded. The energetic value of the crop, the duration of the animat's foraging trip and the energy used by the animat during the trip are all available to the experimenter.

3.2 Energy Model

In this simulation I have used an energy model based on the findings of Hodges (1981). Hodges found that bumblebee queens (*Bombus appositus*) foraging on *Delphinium nelsoni* flowers spent an average of 1.69 seconds flying between plants and an average of 0.71 seconds flying between flowers on the same plant. The bees initially visited the bottom flower on a three flower plant 47% of the time, initially visited the middle flower 45% of the time and started at the top flower 8% of the time. An average bottom flower contained 0.52μl nectar, middle flower 0.40μl, and top 0.23μl. Therefore a bee visiting a flower would find an average of 0.43μl nectar.

A two-molar sucrose solution contains 2.53 cal/μl. Pyke (1980) estimates a 600mg *B. appositus* queen expends 0.0625 cal/sec in flight and 0.00483 cal/sec while feeding from flowers. Feeding time t per flower was estimated by Hodges & Wolf (1981) as a function of the amount of nectar (V) in flowers to be:

$$t = 0.777 + 0.746V - 0.061V^2; \quad V \leq 6.0\mu l \qquad (2)$$

Using this data I designed an energy model for the simulation. Each time-step in the simulation equates to one hundredth of a second of 'bee-time'. The animat when unladen has a mass of 60mg (reflecting the mass or a honeybee worker). I have set the maximum crop size (maximum physical volume that an animat can carry) to 50mg due to measured limits on bees of around 80 to 90% of body mass (Wells & Giacchino, 1968; Wolf & Schmid-Hempel, 1989). Beutler (1950) found that foragers leaving the hive take a nectar load with them to meet the energetic demands of the flight. Heinrich (1979) states that bees have been found to carry a mean of 2.1mg of sugar upon leaving the nest and 27mg upon returning. This is reflected in the model where the carried nectar is the only energy source available to the animat. Therefore, when the crop is empty the animat has no more energy available to it and dies. At each time-step the animat gets taxed for the energy it expends. If the animat is in flight then the energy deducted from it is based on the energy usage of the queen honeybee in flight scaled appropriately. This value is for an unladen bee, however, if the bee is carrying a full crop then its mass may have doubled and the energy deducted will be doubled (Wolf *et al.*, 1989; Heinrich, 1975). The cost of movement is therefore:

$$\frac{m_a + m_c}{m_a} \cdot e \qquad (3)$$

Where m_a is the unladen animat mass, m_c is the crop mass, and e is the energetic cost of movement. The energetic cost to movement is either that of flying or that of feeding. Using the caloric value of nectar, the energetic cost of movement is deducted by 'burning up' the appropriate volume of nectar in the crop. The cost in energetic terms to feeding is fixed within this model. In nature the cost to the bee of feeding is dependent on the temperature. During feeding bees often keep their thoracic temperature at the level required to fly by shivering. When the air temperature is high this costs only a small amount of energy. At low temperatures the cost approaches the energetic cost of flying (Heinrich, 1979).

The time between flowers is dependent on the distribution of flowers and the speed of the animat's movement. The movement speed of the animat is simulated to be up to 15km/hr. This is based on the average honeybee forager flight speed of 11-20km/hr (Heinrich, 1979). The distribution of flowers and the quantity of nectar are varied by the experimenter, though when experimenting 'sensible values' for these variables are similar to those found in the natural system being modeled. Note that the feeding time is completely dependent on the amount of nectar in the flower as per the equation derived for bees above. The unloading time at the nest is likewise based on the average time bees spend inside the hive between foraging visits (3 minutes in simulation, 2-4 minutes for real bees; Heinrich, 1979).

3.3 Dead Reckoning

If current heading with respect to some reference direction and current speed are known, the current position of the animat can be computed as follows:

$$dx = \cos(\alpha_i) v_i \Delta t \qquad dy = \sin(\alpha_i) v_i \Delta t \qquad (4)$$

$$x(t+\Delta t) = x(t) + dx \qquad y(t+\Delta t) = y(t) + dy \qquad (5)$$

α_i is the current heading of the animat, Δt is the time interval between measurements and vi is the average speed of left and right motor.

The current heading can either be measured by referring to a reference direction or dead reckoned. Both methods induce errors; dead reckoning of angles soon becomes very unreliable as the animat performs several turns in a row, due to the compounding effects of error. Using an external reference, on the other hand, removes the accumulative error in angle

measurement and leaves only the linear measurement error. It is therefore more robust.

On returning home, the direction and distance are given by:

$$\gamma = \tan^{-1}\frac{y}{x}, \qquad \zeta = \sqrt{x^2+y^2} \qquad (6)$$

In the simulation the animat dead reckons its position throughout all movements using equation (5). Therefore at all times the animat has an approximate vector pointing to the home position. The constant frame of reference for the heading is a vector indicating due north. When the animat is given a goal position to locate, be it the center of the foraging patch, or the home position after foraging is complete, it attempts to minimize the difference between its current vector and the desired vector. It does this by calculating a difference vector. The animat moves such that it minimizes the difference between its heading and the direction to the goal γ, from equation (6), whilst moving forward. It continues to do this until its difference vector is reduced to within an acceptable limit.

3.4 Flower Choice

The choice of which flower in a patch to visit is a complex decision in bees. I have simplified the problem greatly in this simulation. When the animat arrives at the approximate area for the center of the foraging patch, it initiates a behavior that loops till the animat 'decides' to return to the home position to unload its crop. What constitutes the decision to stop foraging is dependent on the experiment. The behavior of foraging consists of finding a flower to collect from, approaching the flower, and feeding. The flower choice is made using ideal information on the distance from the animat's position to the flowers in the local area, and knowledge of which flowers have previously been foraged. A flower is chosen on the basis of being the closest to the animat, and not being previously collected from on that foraging trip. The heading from the animat to the chosen flower is then minimized whilst traveling forward, eventually leading the animat to the flower.

When the animat is within the bounds of the flower it will start to collect the nectar. This will cost the animat energy at the collecting rate for the duration of its stay on the flower (as from equation 2). Once the animat has collected all the nectar it is free to return home or select another flower. If the mass of the animat has gone above that of its maximum crop load it is assumed to be 'too heavy to take off' and will stay stationary on the flower attempting takeoff, i.e., burning energy at the rate of flight, until its mass drops below the maximum crop load.

The movement of the animat is controlled by a single value corresponding to the required heading h. Equation (7) is used to set the right and left motor speeds rs and ls.

$$-\pi < \Delta h < \pi \qquad rs = 2 - \Delta h \qquad ls = 2 + \Delta h \qquad (7)$$

If the difference between the current heading and the desired heading is between 0 and $+\pi$ the animat moves forward and right, if it is between 0 and $-\pi$ the animat moves forward and left. If the difference between the heading and the desired heading is great then the animat will rotate at a high velocity, if the difference is small the animat will rotate at a low velocity.

3.5 Systematic Search

In section 2.3 I described the geometry of the search pattern of desert ants after Wehner & Srinivasan (1981) and Wehner & Wehner (1986). To summarize, the search pattern consists of a number of loops of ever increasing size starting and ending at or near to where the dead reckoning system suggests the home position should be. In the simulation I have designed a similar search pattern for the animat (See figure 3).

When the animat has minimized its dead reckoned home vector it will most probably be a small distance from the actual home site. Once there the animat selects a direction in which to spiral, either clockwise or anti-clockwise, at random with a probability P=0.5. The animat then heads in a direction 1.6 radians (91.7°) off the dead reckoned vector pointing to the estimated home position. Any value larger than 90° will cause the path to spiral out, any smaller value will cause the path to spiral in. For clockwise rotation the animat attempts to keep its heading at the home vector angle +1.6 radians, for anti-clockwise rotation the heading is kept at the home vector angle -1.6 radians. Initially the animat keeps the heading relative to the home vector for 3 seconds (300 simulator time steps). This duration was chosen by trial and error, making the duration larger would cause the spirals to be larger. If the animat passes within a close distance to the true home site the search process is finished. If, however, the animat has not stumbled upon the home position then it returns to the expected home position (using the dead reckoned home vector) and starts on a second spiraling path. This path may either be in the same direction as the last, or in the opposite direction. Again this is decided using a probability of P=0.5. The whole process of spiraling and returning to the estimated home position is repeated with a spiraling duration increasing by 0.5 seconds (50 time steps) each time. This search pattern will continue until either the animat locates the home position or it runs out of energy.

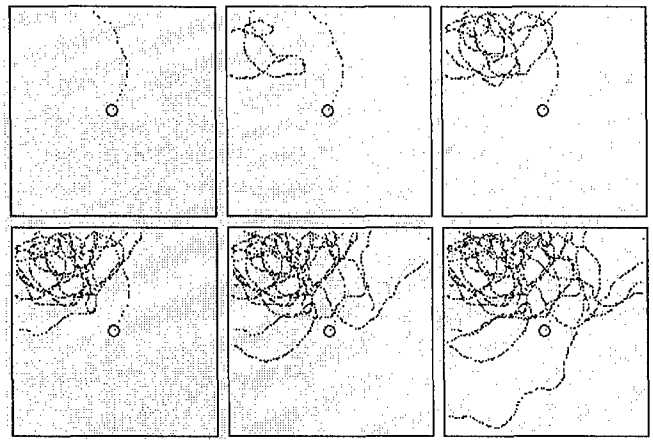


Figure 3: The spiraling path of the animat attempting to locate the home position. The home position is represented by a circle. The path of the animat is shown by the dotted line. The first diagram (top left) shows the animat's path as it left the home position. The second diagram (top center) shows the animat returning to the fictive home position and starting on its spiraling search path. As time progresses the animat's path grows wider, searching larger areas for the home position. By the sixth diagram (bottom right) the home position is within the search path, shortly after this the animat successfully found the home position.

4 Experiment

This experiment is based on basic central place foraging. Here the mechanisms described earlier are employed by the animat in order to successfully increase the energy stores of the nest. One of the decisions that has to be taken in order to maximize efficiency is that of when to stop foraging and return to the nest. As the crop load increases with foraging, so does the mass of the bee. Experiments on free flying honey bees show there is a positive correlation between rate of oxygen consumption and body mass of the bee such that a full load would almost double the bee's energetic needs (Wolf *et al.*, 1989; Heinrich, 1975). Therefore, a bee that has a full crop uses almost twice the energy to move as does a bee with an empty crop. This causes the bee to suffer from a problem of diminishing returns (Krebs & Davies, 1993). As a consequence bees have evolved a strategy for foraging such that bees are often observed returning to the hive with less than the maximum load they can carry (see figure 4).

4.1 Experimental Conditions

In this experiment I evolve the decision of when the animat should return home as the distance from the nest to the foraging patch is increased. In a natural environment it is possible that from one season to the next the distance to the foraging patch changes dramatically due to weather conditions or changing crop patterns. This can result in two to three-fold changes in honey production from one year to the next (Winston, 1987, p.170). A bee or animat's rule of thumb for foraging must encompass such changes since they are frequent. By evolving the strategy for different cases one can see what the optimal strategy is when maximizing energetic efficiency or when maximizing net energetic gain.

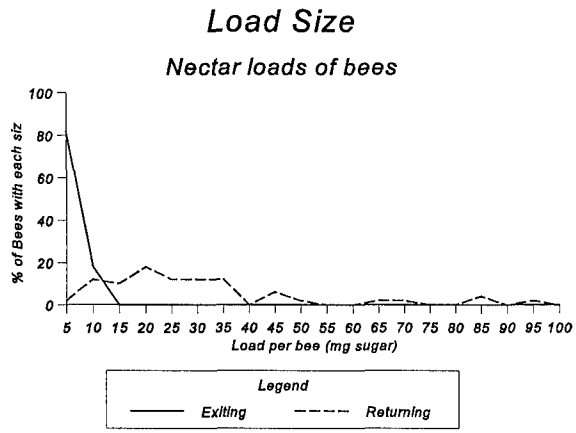


Figure 4: Sizes of nectar loads (in mg sugar) carried by workers entering and leaving a *B. vosnesenskii* nest in Berkeley. (Based on Allen *et. al.*, 1978)

Each of the experiments explained within this section are based on a population of 30 animats. Each animat has a genotype of 16 bits defining both its decision on when to leave the foraging patch and how much nectar to take with it to ensure it has enough energy to reach the patch. In this case the first 8 bits of the genotype define the number of food sources the animat will visit; from 1 through to 256. The second 8 bits define the volume of nectar the animat loads up with for its outward journey; from 1mg to its maximum crop capacity.

Each animat performs a foraging excursion. Upon return to the nest the amount of nectar the animat carries is recorded. This, together with the time the animat took for the excursion and the total energetic expenditure of the animat throughout the trip are used to calculate a fitness for that animat. After evolving the animats for 10 generations the top scoring animat's ultimate load is recorded. The whole process is repeated ten times and the average of the top strategies is plotted on a graph. This process is repeated as the distance to the foraging patch is increased.

This process is carried out twice, once with a fitness function that maximizes the net gain of energy per energetic expenditure, and again with a fitness function that maximizes the net gain of energy per time. At the beginning of the experiment the foraging patch was located around the nest. The width of the foraging patch was 80m, and it contained a total of 1000 food sources. Each food source contained 0.43mg of nectar. The average distance to the foraging patch for honeybees in agricultural areas is a few hundred meters (Visscher & Seeley, 1982), though through experimentation

recruitment to feeding stations can be induced up to 10km (Knaffi, 1953) if no other competing food sources are available. The distance to a foraging patch dramatically alters the economics of foraging, not because of the energy cost of flying to and from the patch but purely due to the time involved (Heinrich, 1979). A bee foraging from a patch 3km from its nest must spend 24 minutes in flight per trip (flying at 15km/h) plus the 10 minutes required to fill its crop. If the bee could forage from a patch close to the nest its net rate of energy gain could be tripled. In this experiment I increased the distance to the foraging patch from zero through 7.2km.

4.2 Results

The results from this experiment are shown in figures 5 and 6. In figure 5 I have plotted the ultimate mass of the animat upon return to the hive, as distance to the foraging patch varies. The graph indicates that when animats maximize energetic profit (gain in energy per time) they consistently return to the hive with a greater load than if they maximize energetic efficiency (gain in energy per expenditure). This difference is most pronounced when the distance to the patch is small.

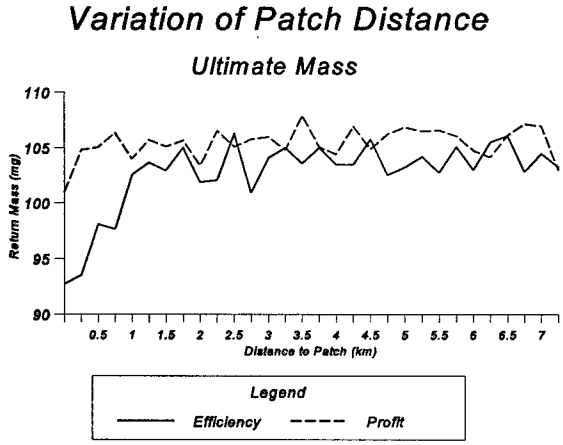


Figure 5: Graph showing animat mass upon return to nest, whilst varying the distance to the center of the flower patch.

Figure 6 shows the number of food sources visited on the foraging trip of the highest scoring animat when the distance to the patch is varied. As you would expect, the number of food sources visited increases when the patch is further away. However when evolved under the fitness function of maximizing energetic efficiency the animat visits less food sources than when evolved using a fitness function that maximizes energetic profit. This is increasingly noticeable the closer the patch.

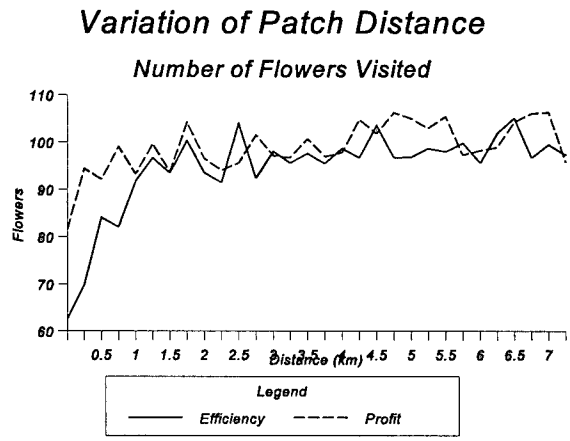


Figure 6: Graph showing number of food sources visited per foraging trip when the distance to the foraging patch is varied.

5 General Discussion

When Schmid-Hempel *et al.* (1985) tested whether the bee's decision about when to leave a foraging patch and return home was influenced by diminishing returns they concluded that it was. They arrived at this conclusion due to the fact that bees would return home with smaller loads when forced to forage on more distant patches, or when the flight time between flowers in the patch was increased. They compared the experimental results with predictions made using mathematical models that maximized either net rate of energy gain per time (profit), and energetic gain per expenditure (efficiency). The model that maximized profit suggested that the bees should return carrying the maximum load possible under all situations. The efficiency model predicted that the bees would sometimes return with smaller loads. They therefore concluded that bees maximize efficiency rather then profit.

In the experiments described in this paper I evolved the decision of when to leave the foraging patch under varying environmental conditions. I found that when the animats were evolved to maximize profit they returned to the nest with an almost full load no matter what the distance to the patch. This result is in agreement with the model of Schmid-Hempel *et al.* However, when the animats were evolved to maximize efficiency I found the results to be very different. Under all circumstances the animats return with a smaller load than when maximizing profit, though in most cases this difference is small. When the patch is close to the nest I find the animats return with the smallest loads. This is in conflict with the findings of Schmid-Hempel *et. al.*.

The fact that bees are often seen to return to the hive with less than a maximum load suggests that they are not maximizing profit. The hypothesis that the bees maximize efficiency is well supported by verbal arguments suggesting

that a survival cost exists such that harder working workers die sooner and thus colony growth is not necessarily maximized by maximizing net rate of energy delivery to the hive. Because of the increased mortality rate associated with such hard work, a portion of incoming resources has to be invested to replace foragers rather than contributing to colony growth. This theory is supported by Neukirch (1982) who explains that foragers have an average life-span of 4-5 days averaging 10 trips per day, though bees tended to die after flying 800km whether this was carried out after fewer visits far away or many that were close. The explanation given is that the bees death is due to the breakdown in enzymatic mechanisms which metabolize carbohydrates in glycogen.

So why do the results from these experiments differ from those of Schmid-Hempel *et al.*? The experiments carried out by Schmid-Hempel *et al.* only tested the foraging decisions of bees at two distances from the hive. Though this data showed that the bees were not maximizing profit it was not exhaustive enough to show what the bees foraging decisions would be at many different patch distances. Figure 4 shows that the majority of bees return to the hive with less than full crop loads. Bees in natural situations have a preference to forage on closer patches (providing the richness of the close food sources are sufficient) than distant patches. It would therefore not be surprising if the small crop load predicted by this simulation holds true when the distance to the foraging patch is small.

However, this does not explain the reduced loads witnessed by Schmid-Hempel *et al.* at distant foraging patches. The verbal argument alluded to earlier in this section suggests that bees do not maximize profit at all, rather the hive attempts to maximize the energetic efficiency, inclusive of the cost of producing the forager. Therefore it would seem reasonable to expect the bees' individual foraging decisions to maximize both its energetic profit and its life expectancy. In a simple case this approximates maximizing efficiency due to the fact that the bee's life expectancy is related to its energetic expenditure. If one now considers the case where the bee is forced to forage from a patch very distant from the hive the situation becomes more complex. While my model suggests the bee that maximizes efficiency should carry back a large load, it ignores the almost certain decrease in life expectancy of the bee. This is due to many factors such as an increased probability of the bee getting lost, a higher risk of running into adverse weather conditions, and an increased risk of predation. Carrying a smaller load would decrease some of these potential threats. Indeed, whilst experimenting, Boch (1956) observed that when a storm was approaching, bees that were foraging on a distant patch (6km from their hive) returned to the hive and ceased foraging activity by 18:09 while bees that foraged on close patch (100m) continued working until 19:15 when the rain began to fall. This suggests that the bees foraging decisions incorporate many more factors than modeled in this or any other system.

6 Future Work

Honeybee colony foragers continually collect food until they die. The efficiency with which they do this affects the amount of energy entering the hive. If the amount of energy entering the hive increases then the rate of growth of the colony increases. Maximum hive fitness is achieved when the hive has the opportunity to grow in size very quickly thus allowing the queen and workers to get down to the reproductive stage of hive development earlier in the season. This allows production of many male bees that then leave the hive to find young queens to impregnate. Therefore the fitness of the hive is largely dependent on the behavior of all the foragers and the workers looking after and producing male bees and young queens. This situation is a lot more complex than the experiments described within this paper.

It is tempting to suggest a simulation of whole hives of animats inspired by the ecology of honeybees. This simulation (though very complex) could be useful in investigations of many aspects of honeybee ecology, from navigational strategies of the individual to the evolution of the dance language.

7 Conclusions

This paper has addressed the simulation of a group of behavioral traits common to many insects that allow them to forage efficiently and reliably. The theory that bee foraging behavior maximizes energetic efficiency rather than energetic gain is based on empirical observations that bees often return to the hive from a foraging trip with a less than full crop. While this experiment supports this view, it also highlighted differences between the observed behavior of bees and the theoretical model. I postulate that the differences are due to the bees attempt to maximize life expectancy as well as energetic profit.

Acknowledgments

This research was funded in part by the States of Jersey Education Committee.

References

Beutler, R. Zeit und Raum im Leben der Sammeldiene. *Naturwissenschaften*, 24:286-291. 1950.

Camhi, J.M. *Neuroethology.* Sinauer Associates. 1984.

Cole. Colony efficiency and the reproductivity effect in *Leptothorax allardycei* (mann), *Insec. Sociaux*, 31, 403-407, 1984.

Goerner, P., & Zeppenfeld, C. The runs of *Pardosa amentata* (*Araneae, Lycosidae*) after removing its cocoon. *Proc. Int. Congr. Arachnol.* 8:243-248. 1980.

Goerner, P., & Claas, B. Homing behavior and orientation in the funnelweb spider, Agelena labyrinthica. In *Neurobiology of Arachnids*, Barth, F.G. (Ed.) , pp. 275-297. Berlin, Heidelberg, New York: Springer. 1985.

Heinrich, B. Thermo regulation by bumblebees II Energetics of warmup and free flight. *Journal of Comparative Physiology* 96, 155-166. 1975.

Heinrich, B. *Bumblebee Economics*, Harvard University Press, Cambridge MA, 1979.

Hodges, C.M. Optimal foraging in bumblebees: Hunting by expectation, *Anim. Behav.* 29, 1166-1171, 1981.

Hodges, C.M., & Wolf. Optimal foraging in bumblebees: why is nectar left behind in flowers? *Behav. Ecol. Sociobiol.* 1981.

Hoffmann, G. The search behavior of the desert isopod Hemilepistus reaumuri as compared with a systematic search. *Behav. Ecol. Sociobiol.* 13:93-106. 1983.

Krebs, J.R., & Davies, N.B. (Eds.) *An Introduction to Behavioural Ecology*(3rd edition). Blackwell Scientific, Oxford, 1993.

Lee & Winston. The effect of swarm size and date of issue on comb construction in newly founded colonies of honeybees (*Apis Mellifera*), *Canadian Journal of Zoology* 63, 524-527, 1987.

Mittelstaedt, H. Analytical cybernetics of spider navigation. In Barth, F.G. (Ed.). *Neurobiology of Arachnids*, Springer, Berlin, 298-316, 1985.

Papi, F. *Animal Homing*, p.1-18, Chapman and Hall, London, 1992.

Pyke, G.H. Optimal foraging in bumblebees : Calculation of net rate of energy intake and optimal patch choice. *Theor. Pop. Biol.* 17, 232-246. 1980.

Saila, S.B., & Shappy, R.A. Random movement and orientation in salmon migration. *J. Cons. Int. Explor. Mer.*, 28, 153-166. 1963.

Schmid-Hempel, P., Kacelnik, A., & Houston, A.I. Heneybees maximize efficiency by not filling their crop. *Behavioural Ecology Sociobiology*, 17, 61-66, 1985.

Seyfarth, E. A., Hergenroeder, R., Ebbes, H., Barth, F. G. Idiothetic orientation of a wandering spider: Compensation of detours and estimates of goal distance. *Behav. Ecol. Sociobiol.* 11:139-148. 1982.

Ugolini, A. Visual information acquired during displacement and initial orientation in *Polistes gallicus* (*Hymenoptera, Vespidae*). *Anim. Behav.* 35:590-595. 1987.

von Frisch, K. *The dance language and orientation of bees*, Oxford University Press, London, 1967.

Waddington, K.D., & Heinrich, B. Patterns of movement and floral choice by foraging bees, in Kamil, A.C. & Sargent T.D. (Eds.). *Foraging Behavior - Ecological, Ethological, and Psychological Approaches*, 215-230, Garland STPM Press, New York NY, 1981.

Wehner, R. Arthropods, In Papi F. *Animal Homing*, pages 45-144, Chapman and Hall, London, 1992.

Wehner, R., & Srinivasan, M.V. Searching behavior of desert ants, genus *Cataglyphis* (*Formicidae, Hymenoptera*). *J. Comp. Physiol.* 142, 315-338. 1981.

Wehner, R., & Wehner, S. Path integration in desert ants, Approaching a long standing puzzle in insect navigation, *Monitore Zool. Ital.*, 20, 309-331, 1986.

Wells, P. H. & Giacchino, J. Relationship between the volume and the sugar concentration of loads carried by honeybees. *Journal of Apicultural Research*, 7:77-82. 1968.

Wilkinson, D.H. The random element in bird 'navigation'. *J. Exp. Biol.*, 29, 532-560. 1952.

Winston, M.L. *The Biology of the honeybee*, Harvard University Press, Cambridge MA, 1987.

Wolf, T.J., & Schmid-Hempel, P. Extra loads and foraging life span in honeybee workers, *Journal of Animal Ecology* 58, 943-954, 1989.

AN EVOLVED FUZZY REACTIVE CONTROL SYSTEM FOR CO-OPERATING AUTONOMOUS ROBOTS

Robert Ghanea-Hercock & David P Barnes

Department of Electronic & Electrical Engineering, University Of Salford, Salford, M5 4WT, U.K
Phone: 0161 745 5000 x3633 Fax: 0161 745 5999
Email: *r.a.ghanea-hercock@eee.salford.ac.uk* or *d.p.barnes@eee.salford.ac.uk*

ABSTRACT

Autonomous robots operating in unstructured environments clearly require some capacity for adaptive behaviour. This paper presents a method for achieving an adaptive response based on a multi layer control architecture. The primary level of control is composed of the Behaviour Synthesis Architecture [1] which is continuously modified by a secondary layer, composed of a Fuzzy Logic controller. The Fuzzy rule base adjusts the relative weighting (utility) of each behaviour to allow the robot to 'focus' its response on the current sensory stimuli. This allows the robot to balance conflicting behaviours and goals in a dynamic environment. The Fuzzy rule base is evolved by genetic algorithms in a simulation of the robot and environment. The advantages of the genetic search algorithm are therefore combined with a set of high level rules and behaviours to resolve the conflicting dynamics of two co-operating mobile robots.

Keywords: Autonomous robots, Applied adaptive and collective behaviour.

1. Introduction

The problems of controlling an autonomous physical agent have been well defined in the field of Alife and studies of adaptive behaviour. Techniques based on reactive or behavioural control [2] have proved successful in achieving real-time responses and allowing robots to interact with their environments without explicit world models in the traditional A.I sense. However they have not proven easy to scale up to cope with more dynamic environments or complex task sequences. Some means of adapting a reactive control system is therefore required which provides better conflict resolution between groups of behaviours. This paper presents a hybrid architecture which attempts to combine the advantages of reactive systems with a 'meta' level of behavioural control generated by a Fuzzy Logic controller. The selection of the fuzzy rule base which determines the adaptive properties of the total system have been evolved by a genetic algorithm [12] in a simulation of our co-operating robots. This is very similar to the approach taken by Ram and Arkin [13], in which genetic algorithms are used to optimise navigation control parameters.

Many have taken the view that to produce fully 'life like' responses, the control architectures should be evolved entirely from low level primitives. We would argue that from an engineering perspective and the goal of creating robots with a high degree of autonomy, it makes more sense to start from a higher level of behavioural capabilities, and to evolve the interactions between these capacities. However this requires that a designer accepts implicitly that any preselected behavioural primitives will restrict the possible evolutionary strategies available to the robot. For the immediate future of practical robotics however, the advantages of a restricted use of evolution in robot control seem to outweigh the limitations it imposes. This philosophy changes the perspective of what constitutes the phenotype and the genotype of an agent. The phenotype is thus best seen as the combination of actuators, effectors and behavioural repertoire while the genotype encodes the pattern of interconnection between these elements, rather than the behaviours themselves.

It is certainly useful to study the process of evolution working from the ground up when dealing with simulated agents or for the study of evolution itself, but when the concern is to engineer animate physical agents with useful properties it makes little sense to start from DNA and hope to reach ants!

"GAs should be used as a method for searching the space of possible adaptations of an existing robot, not as a search through the complete space of robots" [3]

The specific goal of our research group is to enable multiple autonomous robots to co-operate with each other to perform some useful task [4]. Figure 1. shows two of our mobile robots, named 'Fred and Ginger', which can successfully relocate an object while moving through a semi-structured environment. The need to avoid high bandwidth links between such agents, and the problems of encoding and decoding knowledge of the environment are well known, [5]. This resulted in a design which allows simple tactile feedback between the robots via a flexible coupling mounted on their upper structures, (a self-centering x-y table with optical encoders to measure displacement and a freely rotating coupling joint). This work was based on purely reactive control using the Behaviour Synthesis Architecture and encountered the typical problems associated with potential field methods [6], e.g. becoming trapped in local field minima.

Section 2 briefly details the particular reactive control architecture that was used as a basis for the control structure. Section 3 outlines how we combined this reactive system with Fuzzy logic, and section 4 details the genetic fuzzy system used to evolve full control structures.Section 5 presents some initial results with an analysis provided in section 6.

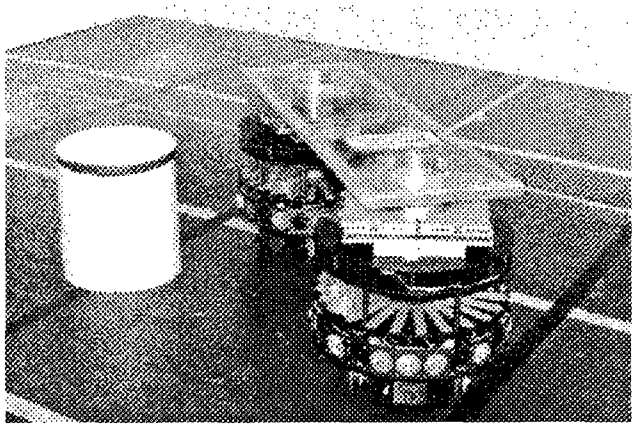


Figure 1. Fred and Ginger, transputer based autonomous robots built on B12 base units, with ultra-sound and infra-red sensors.

2. Behavioural Control

In order to construct a multi-layer control architecture we needed a reactive architecture with a suitable structure, which could provide access to a suitable control parameter which can determine the relative contribution of an individual behaviour. The Behaviour Synthesis Architecture (BSA) is a reactive control mechanism which has been successfully applied to the control of co-operating autonomous devices for an object relocation task [1]. The problems of conflict resolution between behaviours is resolved through a vector synthesis mechanism. Each behaviour pattern is defined in terms of a stimulus/response function. The respective functions have associated stimulus/utility functions, which describe their importance at any particular instance within a given dynamic environment.

$$bp_t = \{\ r_t = f_r(s_t)\}\{u_t = f_u(s_t)\ \} \qquad (1)$$

In eq.1 r_t is the particular motion response at time t and this is a function, f_r, of a given sensory stimulus, s_t. Associated to every response is a measure of its *utility* or importance, u_t. This quantity is a function, f_u, of the same sensory stimulus and the values of r_t and u_t constitute a vector known as a *utilitor*. Competing utilitors are resolved by a process of linear superposition which generates a resultant utilitor for each behaviour pattern , which are then summed to give a final utilitor UX_t. where,

$$UX_t = \sum_{n=1}^{m} u_{t,n} \cdot e^{j \cdot r_{t,n}} \qquad (2)$$

m equals the total number of utilitors, and j is the imaginary complex number. For a resultant utilitor, uX_t is the magnitude of the vector, and rX_t is the argument in the polar plane.

X corresponds to a particular degree of freedom, e.g. translate or rotate, and the resultant motion response, rX_t, is then executed by the robot. From (2), it can be seen that generating a resultant response from different behaviours within the architecture constitutes a process of *additive synthesis*. Figure 3. shows some simple behaviour pattern examples. The utility associated with each behaviour therefore provides an ideal input with which to adaptively modify a behaviour's contribution to the control of the robot. This higher level input is taken from a second control layer which aims to balance the total reponse of the robot in a complex environment. This second control layer is described in section 3.1.

Individual behaviour patterns can be grouped together to form a behaviour packet and sets of these packets are

switched on or off according to sensory pre and post conditions by a sequential script. This provides a mechanism to constrict the conflict between behaviours and allow complex task sequences to be specified, such as navigating to a beacon, collecting an object, taking this to another robot and transfering the object to the second robot. It also allows control over the relative *altruistic* or *egotistic* actions of the agents. A reflective planning agent which can interface to multiple reactive agents is currently being developed and will transmit scripts to the agents via an RF communication link. (At present each script is manually designed). It will also monitor the agents progress in completing a task and send new scripts as required. This will then be linked with an operator interface which would allow a user to graphically direct the robots to perform a task sequence, and the planning agent would generate the appropriate behaviour script for the reactive agents. Details of behaviour scripts and our reflective planning agent can be found in [4].

3. Fuzzy Logic Control

It has been shown that Fuzzy Logic is a useful system for the control of complex systems with incomplete or uncertain information and several groups have successfully applied this method to the control of autonomous robots, [6], [7]. The Fuzzy Logic system normally forms the complete control architecture where each behaviour required, such as obstacle avoidance, is directly encoded into a set of fuzzy rules. The principal advantage is the linguistic basis of encoding the required functions into rules, which facilitates design, and the ability to analyse the Fuzzy rule matrix once created. It also permits multiple input and output dimensions. A Fuzzy Logic controller works by encoding an experts knowledge into a set of rules which are smoothly interpolated and the resultant is defuzzified to give a crisp actuation output. Each rule is specified as either a trapezoid or some other function e.g gaussian and assigned to some range of input variable, see figure 2.

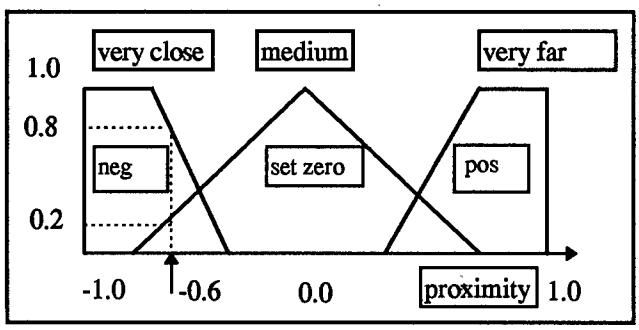


Figure 2. Typical assignment of obstacle distance range values onto a set of fuzzy membership functions, for directly controlling a mobile robot. The scaled input range value -0.6 is a member of the set Negative to degree 0.8 and the set Zero to degree 0.2.

A convenient method of storing and representing fuzzy rules is with a fuzzy associative memory matrix, (FAM) [8]. The base dimensions represent the input variables and each FAM matrix entry represents an output fuzzy set. There are typically 3 to 5 output membership sets, e.g negative large (NL), negative small (NS), zero (ZE), positive small (PS), and positive large (PL). There are many methods of defuzzifying the output from a fuzzy system, normally based around computing the centroid values for the overlapping output membership functions, but in this work we take the simplified case of assigning a singular discrete value to each output membership set. The weight for the i_{th} FAM entry was calculated using the minimum rule:

$$w_i = \min\{F_{(i)}(x), F_{(i)}(y)\} \tag{3}$$

$$B_i = \text{output fuzzy set} \tag{4}$$

The total defuzzified response for n output membership sets is then:

$$F_T = \frac{(\sum_i w_i \cdot B_i)}{(\sum_{i=1}^{n} w_i)} \tag{5}$$

3.1 Combining Reactive and Fuzzy Control

Behaviour based methods, such as Subsumption [2], and BSA offer an intuitive and direct mapping of sensory stimuli to response which is well suited to mobile robot control. As stated previously the BSA method provides a direct means of adapting the relative utility of each behaviour from a higher level control layer, hence it has been used for the purpose of this research. In order to utilise the benefits of fuzzy and reactive systems a layered approach has been implemented; see figure 3.

The fuzzy control layer takes direct sensory input and applies negative feedback on the utility of selected behaviours. The contribution of each behaviour is therefore adjusted dynamically to suit the current set of environmental stimuli. The effect is to allow the robot to focus on its current response while maintaining some awareness of a more general goal. An example of this is when the robot enters a potential well while navigating towards a beacon [11]. As the robot approaches the obstacle, the importance of avoiding it increases due to the utility - response generated by the BSA system, while the fuzzy rule base responds by turning down the utility of moving towards the beacon.

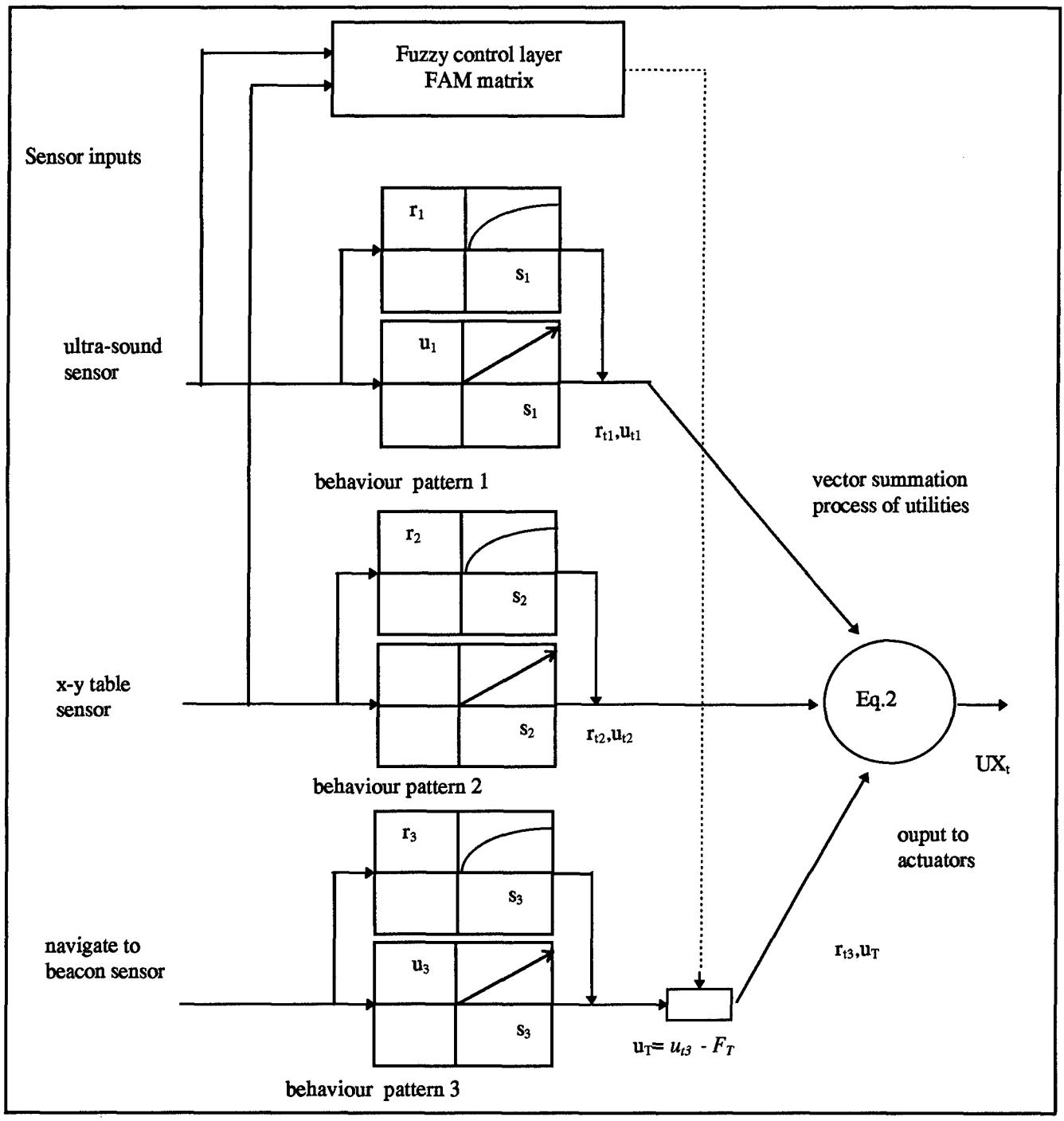


Figure 3. Layered control system showing relationship between fuzzy system and behaviours inside BSA. In this example the FAM matrix is controlling the utility of moving towards an infra-red beacon in response to inputs from the ultra-sound and robot x-y table sensor.

A small bias on the rotate right behaviour breaks any symmetry of the environment, and the emergent response is for the robot to follow the perimeter of the obstacle until a free path towards the beacon is found when it returns to navigating in that direction. A major advantage of this method is that the utilities of multiple conflicting behaviours can be dynamically modified prior to generating a resultant utilitor, by increasing the number of input/output dimensions of the fuzzy rule base, without modifying the set of behaviour patterns themselves.

4.Genetic Fuzzy Systems

The principal difficulty in using a fuzzy system is the rapid increase in possible combinations of rules along with the number of input and output dimensions. This has lead many researchers to use global search algorithms and artificial neural networks to find an optimum set of rules. We have applied a standard genetic algorithm [12], to the process of selecting the FAM matrix entries, with manually selected fuzzy input sets and functions. The encoding scheme for genetic optimisation takes each FAM matrix entry and assigns it to a binary vector, these are then strung together to form a single fixed length chromosome vector. The GA uses a select fittest mechanism and mutation probability was approximately 0.02.

In this work each FAM entry is represented by a 3 bit vector giving 8 possible output fuzzy sets. With N inputs and M fuzzy sets per input the total chromosome has M^N x 3 bits, which for M = 3 and N = 2 gives a chromosome vector of 27. However this will increase rapidly as the number of inputs or sets per input increase, which implies an obvious increase in computational cost during the evolution phase. However due to the modular nature of FAM systems, several could be evolved separately to solve particular aspects of the environment before being combined into a final system. Each complete FAM matrix is then evaluated against a specified fitness function and the normal processes of cross-over and mutation applied. In this case the fitness function is a combination of avoiding obstacles and achieving a minimum displacement of the x-y table, see fig.1, which links the robots together. The fitness function is a simple product of distance traveled towards the beacon and average displacement of the x-y table. The general form of an optimum FAM matrix, in this example, is therefore a positive gradient in these planes, although the response of the robots to the specific environment may significantly modify the detail within the action surface as the total system is highly non-linear.

4.1 Implementation

A dynamic simulation of the coupled mobile robots has been created which runs the BSA and fuzzy control system. Each robot has 5 forward looking obstacle sensors, an internal measurement of the displacement of its coupling x-y table to the other robot and a global navigation sensor to detect the target beacon. This is implemented on a 486 PC and typical times to evolve a working FAM matrix is four to eight hours, depending on population size (where each individual encodes a single FAM matrix) and complexity of environment.

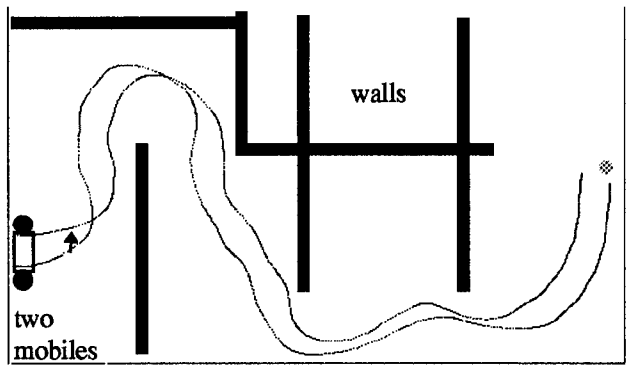


Figure 4. Typical path of coupled robots using an evolved control structure.

5. Results

The effect of increasing population size on the 3D FAM matrix is shown in the following figures. In figure 5. the small population of 24 restricted the system to a sub-optimal solution, where the applied feedback on the distance to obstacle behaviour utility is insufficient to avoid collisions.

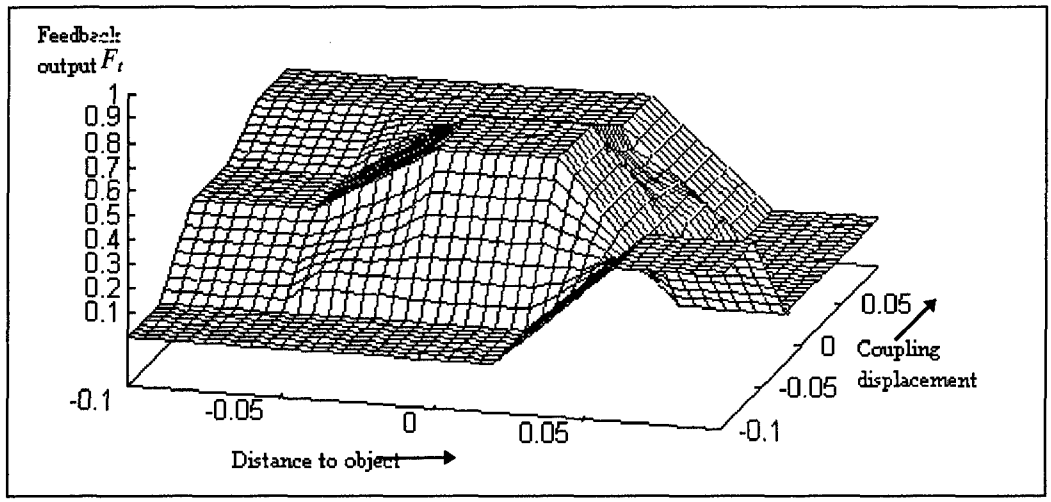


Figure 5. Fuzzy action surface with population size of 24. As either the distance to object (scaled as -0.1 = max distance from obstacles and 0.1 = min distance, for the FAM matrix) or coupling displacement increase (-0.1 = min x-y table displacement, 0.1 = max displacement), the FAM matrix output should increase to reduce the utility of any conflicting behaviour such as the navigate to beacon behaviour.

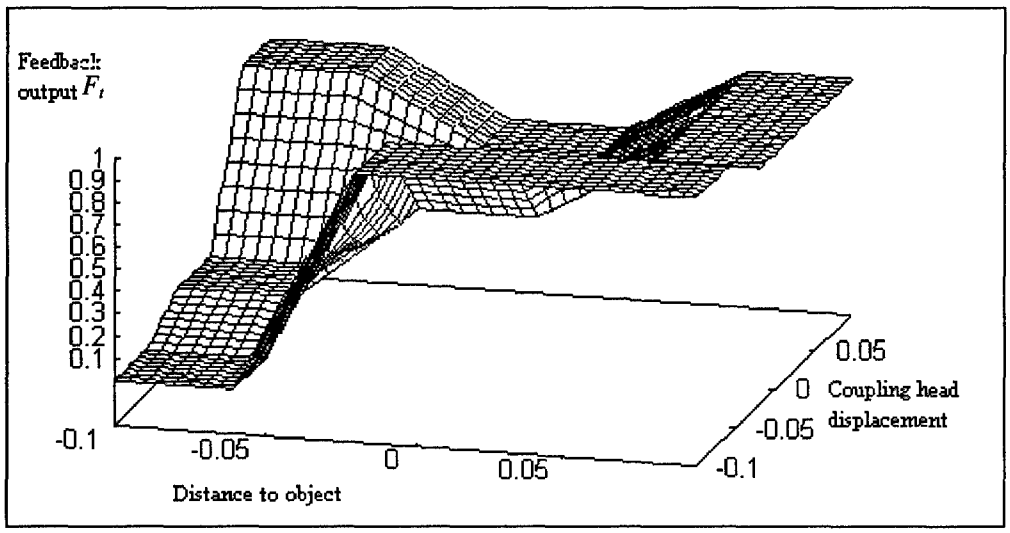


Figure 6. Fuzzy action surface with population size of 44. In this example the surface applies maximum feedback when either input exceeds approximately 20% of the possible range. This allows the robots to reach the target but gives poor control over the coupling between them.

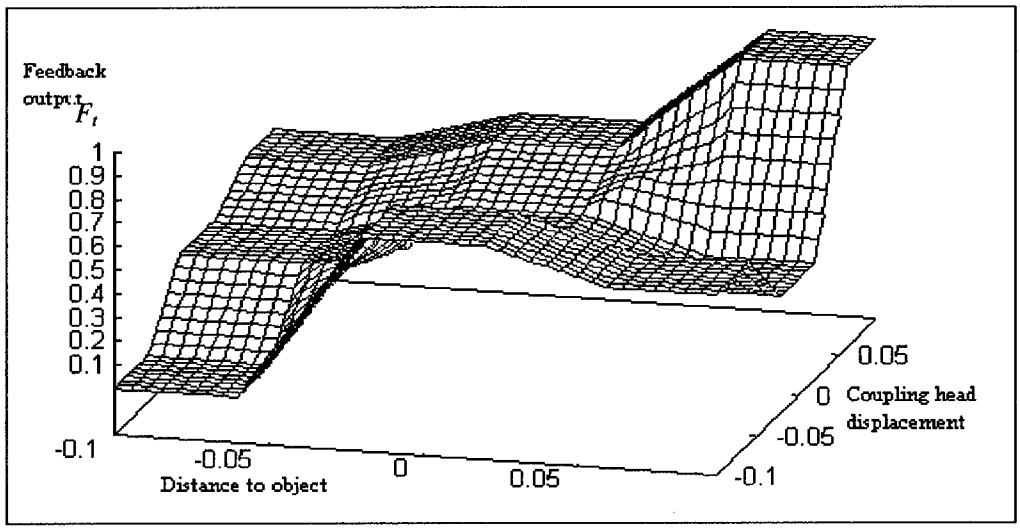


Figure 7. Fuzzy action surface with population size of 66.

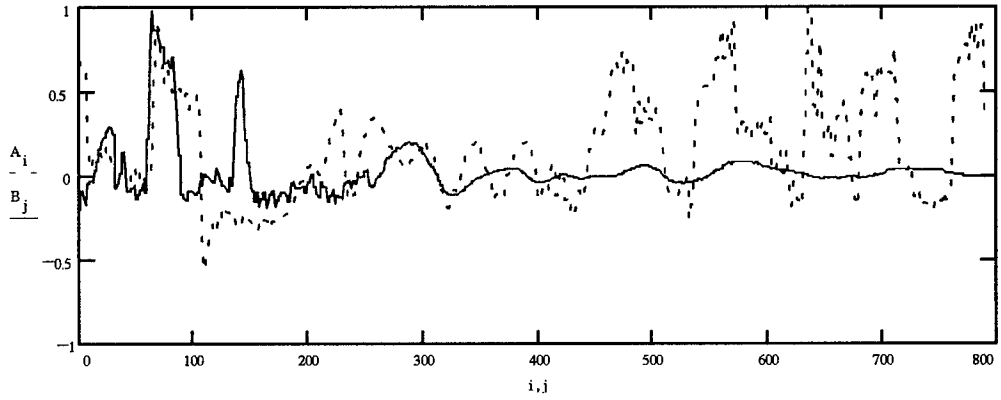


Figure 8. Plot of capture head oscillation during runs with evolved controllers: dotted line A_i = population of 44, solid line B_j = population of 66.

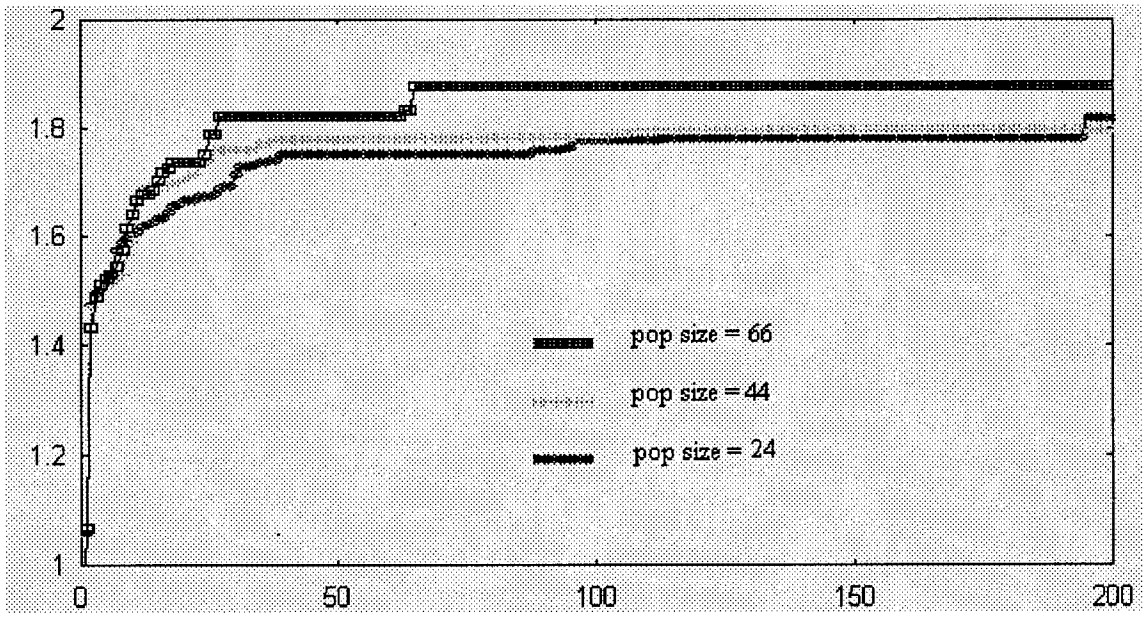


Figure 9. Best fitness values for three population sizes over 200 generations..

With population sizes above 30 - 40 the mobiles begin to reach the target and avoid colliding with the walls. Comparing the plots in figures 6 and 7 with the actual runs showed that even with quite irregular action surfaces the robots still smoothly navigated the environment, indicating that the fuzzy control layer is quite robust.

During the first quarter of figure 8. both systems are still negotiating the walled area of the environment, but as they leave this area and move towards the goal, the more evolved solution quickly damps any head oscillation, as specified by the fitness function.

In fig 9. the discrete step increases in fitness reflects the fitness function which rewards heavily for distance travelled towards the beacon, and this is prevented by several difficult corners in the path of the robots. Hence the system reaches a plateau whenever its rule base is insufficient to overcome a particular corner.

6. Analysis

This work has attempted to combine the advantages of three methods of robot control and apply the best strengths of each to solve the difficult problem of autonomous agent design. From the results achieved so far we can infer several points about the combination of Fuzzy and Reactive control techniques.

The addition of a second control layer alleviates some of the major failings of purely reactive systems, such as allowing the agent to escape from local potential minima, while retaining their advantages, such as speed of response to stimuli. Without the fuzzy control layer the robots simply cannot negotiate an environment with any potential minima. Secondly by evolving only the parameters within a lower control system relatively small population sizes can give working results, compared to evolving complete control architectures [13].

In a layered control architecture the higher layers should suppress or inhibit lower levels but not subsume them completely. This has the effect of allowing a residual awareness to remain of the suppressed behaviour, which in this case keeps the robot orientated towards the beacon while avoiding objects.

The evolved fuzzy action surface represents and encodes a global memory of interactions between the agent and its environment. It may therefore represent a meta level of behaviour management which forms a 'long-term' memory of how to successfully adapt current behaviours the learnt environment. A multi-layer approach to behavioural control introduces a context dependency to the learning of behaviour interactions and the use of correct stimuli-response functions which may be supported by real animal behaviour:

"*Reflex-like control structures are shown to be overtly simplistic to account for animal behaviour, rather, animal control systems appear to employ hierarchical control structures*" [10].

7.Conclusion

By combining the best features of a reactive system, and a fuzzy controller , with an evolutionary search method, a means of effectively controlling multiple co-operating autonomous robots has been achieved. The application of Alife techniques to autonomous robot control can generate solutions to the complex non-linear control problems in this area, and a hierarchical control strategy can imbue a useful measure of adaptive capability into such systems. We aim to transfer the evolved controller onto the physical robots in the near future. This should be easily realisable as the evolved structure is only controlling the interactions between proven groups of behaviours which have successfully controlled the physical robots before. Also the concept of dynamically coupling groups of behaviours has been demonstrated in previous work on our mobile robots, [9].

Acknowledgements

Thanks to Dave Eustace for the use of his simulation code and Nazila Ghanea-Hercock for her support. This work was funded by the EPSRC (grant : REF No. GR/J49785). Additional code was used from the text "Neural Network and Fuzzy Logic Applications in C/C++" Welstead S.T. Wiley pub. 1994.

References

1. Barnes, D.P. and Gray, J.O., 1991, "Behaviour Synthesis for Co-operant Mobile Robot Control", Proc. IEE Int. Conference on Control 91, Vol. 2, pp 1135 - 1140.

2. Brooks, R.A., 1986, " A Robust Layered Control system for a Mobile Robot", IEEE Journal of Robotics and Automation. RA-2, No.1, pp14-23.

3. Husbands .P, Harvey .I and Cliff .D "An evolutionary approach to situated ai", Prospects for AI, Proc AISB93, Birmingham England, pp. 61-70 1993.

4. Aylett .R, Coddington .A, Barnes .D, Ghanea-Hercock .R, "A Hybrid Multi-Agent System for Complex Task Achievement by Multiple Co-operating Robots", Multi-Agent Workshop, Oxford 1995.

5. Beckers .R, Holland .O.E, and Deneuberg .J.L, "From Local Actions to Global Tasks: Stigmergy and Collective Robotics", Proc. of the Fourth Int. Workshop on the Synthesis and Simulation of Living Systems, Boston 1994.

6. Watanabe, Y., Pin. F.G, 1993 "Sensor based navigation of a mobile robot using automatically constructed fuzzy rules", Proceedings of ICAR '93 the International Conference on Advanced Robotics, Tokyo Japan, pp 81-87.

7. Surmann .H, Peters .L, Huser .J, "A Fuzzy System for Realtime Navigation of Mobile Robots", 19th Annual German Conf. on AI , KI-95, pp. 170-172.Bielefeld.

8. Kosko .B, "Neural Networks and Fuzzy Systems, A Dynamical Systems Approach to Machine Intelligence", Prentice Hall Int. 1992.

9. Ghanea-Hercock .R, Barnes .D, "Coupled Behaviours in the Reactive Control of Cooperating Mobile Robots", International Journal of Advanced Robotics, Japan, special issue on Learning and Behaviours in Robotics, (accepted for pub. April 1996.)

10. Roitblat .H, "Mechanism and Process in Animal Behaviour: Models of animals, animals as models", Proc. of the Third Int. Conf. on Simulation of Adaptive Behaviour, Brighton 1994, pp. 12-21.

11. Koren Y.,Borenstein J.,"Potential Field Methods and Their Inherent Limitations for Mobile Robot Navigation", 1991 IEEE Int. Conf. on Robotics and Automation, Sacramento pp. 1398-1404.

12. Holland J.,"Adaptation in Natural and Artificial Systems", MIT Press 1992 2nd Ed..

13. Ashwin R., Arkin R., Boone G., Pearce M., "Using Genetic Algorithms to Learn Reactive Control Parameters for Autonomous Robotic Navigation", Adaptive Behaviour, vol.2, issue 3, pages 277-304, 1994.

On Simulating the Evolution of Communication

Jason Noble and Dave Cliff
School of Cognitive and Computing Sciences
University of Sussex
BRIGHTON BN1 9QH, U.K.
jasonn@cogs.susx.ac.uk; davec@cogs.susx.ac.uk

Abstract

The prospects for modelling the evolution of communication are considered, including the problem of intentional explanation, and the possibility of grounding simulation work in theoretical biology. The seminal work of MacLennan and Burghardt [16] on the evolution of cooperative communication is described, and their experiment replicated. Our results were broadly similar, in that evolved communication was observed, but specific differences are discussed. MacLennan and Burghardt's work is extended and their methodology critiqued in detail. Their experiment remains a useful demonstration, but artefactual features make their results difficult to interpret. Furthermore, we argue that too many factors are simultaneously investigated for any general principles to be extracted, and suggest an alternative program of narrowly-focused simulations.

1 Modelling Communication

What has to be happening such that we would describe two entities within a computer simulation as communicating? Can communication behaviour evolve in simulation? Do we learn anything about animal communication or the development of language from such work? MacLennan and Burghardt [16, p. 186] conclude that "even in [a] simple synthetic world, communication may evolve that exhibits some of the richness of natural communication." The intent of this paper is to investigate the prospects for modelling communication through a critical replication of their research.

1.1 Definitions

Communication can be a slippery subject, partly due to the enormous variety of signalling behaviours in nature. Looking at intra-species communication alone we find aggregational signals, alarm signals, food signals, territorial and aggressive signals, appeasement signals, courtship and mating signals, and signalling between parents and offspring [14] — the list is not exhaustive. It can be difficult to capture all this under one definition, or under one explanatory story. Krebs and Dawkins [13] suggest that communication arises because all animals need to be good at predicting the behaviour of other animals in their environment, and that such mind-reading abilities can be exploited, through signalling, to manipulate the behaviour of the receiver. That is, communication evolves when it is beneficial to have one's behavioural intentions predicted (or falsely predicted). Some authors [7] have suggested that this places too much emphasis on dishonest communication, and a typical definition from the biology literature [14] is as follows:

> ...the transmission of a signal or signals between two or more organisms where selection has favoured both the production and the reception of the signal(s).

Note that the role of natural selection is central, and assumed as a premise (see sections 1.4 and 2.1).

1.2 Intentionality

The difficulty of defining communication leads us to the problem of intentionality. In cases of communication between animals or simulated animals, to what extent should we talk about the sender and the receiver as rational, intentional agents? Intentional (and teleological) explanations have always been risky practice in science. However, even the most rigorous accounts of communication (e.g. Shannon and Weaver [21]) are suggestive of intentionality in both senses: an agent who *intends* to communicate something to another agent, via a signal that *means* something. If we consider a phenomenon such as predator alarm calls in vervet monkeys [20], we are a long way from being able to explain this at the level of neurology, and unimaginably far from being able to explain it at the level of physics. However, we can talk usefully about what a monkey intended to achieve by calling, and what a particular call means — indeed, Harré [11] describes primatologists who eschew such intentional language, in the name of objectivity, but end up surreptitiously using it anyway.

Dennett [5] argues that we should not resist the temptation to adopt "the intentional stance" towards complex systems like vervet monkeys. He suggests that this is an example of a more general strategy in science of "changing levels of explanation and description in order to gain

access to greater predictive power or generality" [5, p. 239], and compares it to the abstraction of the concept of *food* in biology. The underlying details of physics, chemistry or whatever are legitimately ignored because they do not contribute to the usefulness of explanatory accounts at the higher level.

So where do we draw the line? Can we ascribe intentions and meaning to just anything? Dennett suggests that there will be no clear division between cases where the intentional stance is appropriate and cases where it is not. As we move to less and less complex systems, mechanistic accounts of their function will become progressively more plausible.

We are not making the bold claim that *animals have intentions.* There is insufficient evidence at this time to support either that claim or its negation. What we are saying is that the concept of communication is an intentional-level concept, and that therefore any attempt to investigate communication without using an intentional framework will be incoherent.

1.3 *Work in theoretical biology, ethology, and behavioural ecology*

There exists an extensive empirical and theoretical literature on animal communication. Empirical work tends to describe a particular type of signalling within one species or between two species (see e.g. [10, 13, 14] for reviews). Some of the most promising theoretical approaches (e.g. Maynard Smith [17]) involve positing a functional model of a signalling system and investigating the conditions under which that system would be evolutionarily stable. The argument is that we should only expect to find relatively stable systems in nature, and that if we are confronted with a system that appears to be otherwise, we should expect to find stability under the surface.

An example of this sort of work is that of Zahavi [25, 26], later validated mathematically by Grafen [7]. They argue that signalling in sexual selection will tend to be honest, and the signals costly. Assume that males signal their reproductive fitness to females through some phenotypic trait: whereas we might expect males to exaggerate their quality, females will be selected on the basis of their ability to discriminate between high and low quality males. The evolutionarily stable situation will be one where a signal is expensive for the male to produce (e.g. the peacock's tail). Thus deceptive communication becomes impossible: if the male can afford to display that signal, he really is of high quality. Through logical and mathematical argument, Zahavi's counter-intuitive idea — that sexual selection can be selection for a handicap — is established.

1.4 *Grounding simulation*

We concur with Miller [19] that such work in theoretical biology and related fields is the best starting point for those who wish to model communication and other biological phenomena *in silico.* While the simulation of adaptive behaviour (SAB) and artificial life (AL) research communities have had prima facie success at simulating evolution, or, if you prefer, generating real evolution inside a computer, it is not yet clear that their approach is a reliable route to general laws or principles concerning animal behaviour. Too often, SAB/AL work falls into the trap of circularity: a simulation is tailored to evolve a specific phenomenon, and that very phenomenon is reported as an unexpected and interesting result.

The notion that evolution is a kind of optimising or satisficing process is not really under debate (*pace* Gould and Lewontin [6]), and SAB/AL work should not be in the business of merely showing that evolution works. In all but the most open-ended simulations, the optimising principle is built in anyway: high scorers on a pre-given fitness function are more likely to reproduce. If SAB/AL is to go forward as a method of scientific investigation, "proof of concept" displays must give way to research that looks for structures underlying the observed phenomena. As Maynard Smith [18] says of the theoretical biology literature, "in using optimisation, we are not trying to confirm (or refute) the hypothesis that animals always optimise; we are trying to understand the selective forces that shaped their behaviour."

There are some recent examples of SAB/AL work on communication that start with a model or theory from biology, perhaps expressed mathematically, then validate and extend that model using iterative, computational techniques. For example, de Bourcier and Wheeler [4] look at aggressive signalling and territoriality. They state that their method of *synthetic behavioural ecology* "is pitched at an intermediate level between, on the one hand, abstract theories based on mathematical models and, on the other hand, empirical observations in complex environments" [4, p. 464].

The two seminal examples of attempts to evolve communication, however, are Werner and Dyer [22] — who simulated the evolution of a simple communication protocol that allowed immobile females to guide blind males towards them for mating — and MacLennan and Burghardt [16]. We feel that both of these papers are worthy of careful reconsideration, but for reasons of space we concentrate entirely on the latter.

2 MacLennan and Burghardt's experiment

2.1 *Justification*

MacLennan and Burghardt describe their method as *synthetic ethology*, contrasting it explicitly with simulation. They state that:

> Our goal in these experiments was to design a synthetic world that was as simple as possible while still permitting communication to evolve. [16, p. 165]

MacLennan and Burghardt repeatedly emphasise that their "synthetic world" is not supposed to reflect any real environment, nor are their simulated organisms like any actual species. Inspired by the synthetic psychology of Braitenberg [2], they hoped that, in comparison with empirical ethology, their stripped-down approach would be "more likely to suggest behavioral laws of great generality" [16, p. 163].

MacLennan and Burghardt were aware of the difficulty of defining communication, and of the problem of imputing intentionality. They adopted Burghardt's [3] definition of communication, which "finessed the issue of intent by the requirement that the behavior be likely to influence the receiver in a way that benefits, in a probabilistic manner, the signaler or some group of which it is a member" [16, p. 163].

They chose to investigate cooperative communication. Possibly this choice was influenced by Burghardt's definition, in that cooperative communication generally benefits both the sender *and* the group to which it belongs. They reasoned that for communication to be selected for, some of the simulated organisms must have access to information that the others in the group did not — otherwise communication would be unnecessary. The non-shared information must also be of environmental significance; it must be worth talking about. In line with their definition of communication, they designed the synthetic world such that communicating this non-shared information would tend to confer a selective advantage.

2.2 *Method*

MacLennan and Burghardt[1] used populations of simulated organisms that they refer to as "simorgs". The simorgs all have access to a shared global environment, and each individual has access to a private local environment. The global environment provides a medium for communication, and the local environments are a source of significant information that the simorgs may evolve to communicate *about*. Each of the environments is as simple as possible, represented by a single variable that can take on a finite number of values. It is emphasised that "there are no geometrical relations among [the simorgs]...they are not in a rectangular grid, nor are some closer than others" [16, p. 166].

MacLennan and Burghardt suggest, by way of analogy, that the global environment can be thought of as the air, capable of transmitting only one sound at a time, and the local environments can be considered exclusive hunting grounds, into which different species of prey may wander. In other words, states of the global environment have the potential to be exploited as signals, and states of the local environment are particular circumstances that it will pay simorgs to signal about.

Simorgs have only two classes of behavioural choice open to them: they can *emit* a signal (into the global environment), or they can *act* in an attempt to respond to the signal of another. The state of the global environment can be changed by any of the simorgs if that simorg emits a signal when its turn comes; the states of the simorgs' local environments are not under their control, and are periodically reset to random states.

In the synthetic world, simorgs achieve fitness by successfully cooperating with another simorg. Specifically, by responding to a signal with an action that matches the local environment state of the signaller. When this occurs, both the signaller and the respondent are rewarded with a point of fitness. Continuing their analogy, MacLennan and Burghardt suggest that this is to be regarded as two hunters bringing down a prey animal that neither could bag alone. Assuming that successful communication has taken place, note that the signal does not mean "I've got some prey here", but "I've got prey of type λ here; would you mind helping out with action-λ?" The state of another simorg's local environment is not directly knowable, and successful cooperation can only come about through a lucky guess or the employment of communication.

In order to implement their ideas in a computer program, MacLennan and Burghardt had to make a number of somewhat arbitrary practical decisions. Thus, time in the synthetic world is discrete. Once each time step, the simorgs respond (i.e., act or emit) in a fixed order; effectively they are arranged in a ring. The program keeps track of the "owner" of the symbol currently occupying the global environment. It is possible, for example, for one simorg to emit and then earn several fitness points consecutively as a series of other simorgs act in response to the same persistent signal.

Every five time steps (one environment cycle) the local environments are reset to a random value, ensuring that the simorgs must react to changing circumstances if they are to succeed. Every fifty time steps there is a breeding cycle: two fit simorgs are stochastically selected as parents and, using two-point crossover with a

[1] MacLennan and Burghardt's methodology is difficult to describe briefly, and the reader is referred to their work [16] and MacLennan's earlier article [15] for a complete account.

small chance of mutation, a new simorg is generated. An unfit simorg is stochastically selected to be replaced by the child, keeping the population size constant. This arrangement is akin to the steady-state variety of genetic algorithms.

The experiments reported were run for 5000 breeding cycles, populations were of size 100, there were eight local environment states (L) and eight global environment states (G) — "just enough possible sounds to describe the possible situations" [16, p. 175] — and the mutation rate was a 0.01 probability of one mutated allelle per birth.

Finite state machines (FSMs) serve as the internal architecture of the simorgs. MacLennan and Burghardt could have used any number of architectures, and considered using neural networks, but settled on FSMs because they "are both readily understood intuitively and easy to represent in genetic strings for simulated evolution" [16, p. 167]. In the experiment described[2] the FSMs were of only one state, which reduces to a look-up table. The response a simorg would make at any one time step was completely determined by the state of the global environment and the state of its local environment. The content of each of the 64 (8×8) entries of the look-up table was a flag indicating *act* or *emit*, and an integer representing the action-type or the emitted symbol respectively. The genetic coding of the simorg was a direct mapping of this structure.

MacLennan and Burghardt included in the program a mechanism to (optionally) prevent communication from occurring: the global environment could be overwritten with a random symbol after the response of each simorg. They reasoned that if fitness increased more rapidly when communication was permitted, compared with when it was blocked, then "true communication (involving a sender)" [16, p. 172] was taking place. In a similar fashion they were interested in exploring the effect of a simple learning rule, whereby a simorg that makes an incorrect action (i.e. an action that does not correspond to the local environment state of the last emitter) in response to a signal has the appropriate entry in its look-up table altered so that it *would have* given the correct response. Thus, they report the results of subjecting the same randomly generated initial population to each of the following experimental conditions:

C^-L^- communication blocked and learning disabled;

C^+L^- communication permitted and learning disabled;

C^+L^+ communication permitted and learning enabled.

[2]MacLennan and Burghardt actually conducted two experiments; we focus entirely on the first. Experiment 2 was an attempt to evolve multiple-symbol communication and the results led them to conclude that "making the step to multiple-symbol syntax is evolutionarily hard" [16, p. 183].

In each of the conditions, they collected data on mean fitness over time. They also constructed a "denotation matrix", which recorded the number of successful communication events, arranged in a table by local and global environment states. They found that these matrices were most useful when tallied over the last 50 breeding cycles of a 5000-cycle experimental run. Under these circumstances, the matrix was interpreted by MacLennan and Burghardt as describing the evolved language of the simorgs. The degree of structure present in the matrix was indexed by co-efficient of variation and entropy statistics.

2.3 Results and conclusions

MacLennan and Burghardt report that communication did indeed evolve in the synthetic world. The results reported are for a single random initial population subjected to each of the three conditions; MacLennan and Burghardt assure us that these results are typical. In the C^-L^- condition, there was only a very slight increase in fitness over the length of an experimental run, whereas in the C^+L^- condition the rate of fitness increase was an order of magnitude greater. In the C^+L^+ condition, the rate of fitness increase was higher still. MacLennan and Burghardt conclude that, when it is not suppressed, communication is selected for and leads to higher levels of cooperation. The provision of the single case learning rule further increases the effectiveness of the communicative strategy.

Analyses of the denotation matrices showed that in the C^-L^- condition, the pattern of symbol use was almost random. When communication was permitted the matrices were quite structured, as measured by the entropy statistic. Visual inspection of the denotation matrices made it clear that certain symbols had evolved to (almost uniquely) represent certain local states. There was ambiguity in two senses: sometimes a symbol would be used to represent two or more states, and sometimes a state was represented by two or more symbols. MacLennnan and Burghardt suggest that the ambiguity is either due to two subpopulations using different symbol dialects, or to individual simorgs using one symbol to represent two different states.

That there should be any fitness increase at all in the C^-L^- condition is not obvious. MacLennan and Burghardt refer to this phenomenon as "partial cooperation through co-adaptation", and regard it as a "low-level effect" [16, p. 185]. They explain it by noting that simorgs can do better than chance if they emit a symbol only in a subset of their local situations, and guess actions within that same subset (see section 4.4 for details).

	MacLennan & Burghardt	Our results *Mean*	*SD*	*p*
Fitness increase				
C^-L^-	0.37	0.99	1.16	n.s.
C^+L^-	9.72	14.6	6.54	n.s.
C^+L^+	37.1	10.6	10.6	0.025
Final mean fitness				
C^-L^-	≈ 6.6	6.74	0.43	n.s.
C^+L^-	10.28	12.71	2.68	n.s.
C^+L^+	59.84	46.13	4.02	0.004

Table 1: Rate of fitness increase (determined by linear regression and measured in units $\times 10^{-4}$ breeding cycles) and final mean fitness scores. Note that mean fitness data was a moving average smoothed over 50 breeding cycles, and that final mean fitness in the C^+L^+ condition is much higher because the simorgs had four chances per environment cycle to respond after correction by the learning rule: fitness scores in this condition start at 40+ rather than the usual chance level of 6.25. Rates of increase are thus a better comparison across conditions.

3 Replication of MacLennan and Burghardt

We replicated MacLennan and Burghardt's experiment, writing our own code[3] based on the published descriptions of their procedure [15, 16].

The replication gave qualitatively similar results, in that fitness improved over time when communication was enabled, and structure developed in the denotation matrices, but the specific results in the three conditions were not reproduced. Table 1 contrasts MacLennan and Burghardt's results with our own; the rate of fitness increase per 10^4 breeding cycles and the mean final fitness are shown. MacLennan and Burghardt's results are taken directly from [16], and refer to the single run they presented as the typical case. Our own results show the mean and standard deviation across 20 runs with different random seed values. For each condition, the column labelled "p" shows the statistical significance of a two-sample t-test of the null hypothesis that MacLennan and Burghardt's result could have come from the same distribution as our data ("n.s." means not significant, i.e. $p > 0.05$).

The C^-L^- and C^+L^- conditions showed slightly higher rates of fitness increase in our own experiment. More importantly, the rate of fitness increase in the C^+L^+ condition was more than three times *smaller* in our data than in MacLennan and Burghardt's, and this was statistically significant. Our results do not support their finding that the C^+L^+ condition, i.e. communication with learning, leads to the highest rate of fitness increase. We found communication with learning to be inferior to communication alone.

[3]Each author worked independently, to allow cross-checking. Both authors used the C language; JN's version was approx. 1500 lines long (including code for various statistics not mentioned here), and DC's version was 560 lines long. A 5000 cycle C^+L^- condition ran in about 80 seconds on a Sun Sparc 20 (both versions). All source code is available on request.

	MacLennan & Burghardt	Our results *Mean*	*SD*	*p*
C^-L^-	5.66	4.96	0.15	< 0.001
C^+L^-	3.95	3.36	0.50	n.s.
C^+L^+	3.47	4.45	0.36	0.015

Table 2: Entropy statistics, calculated on the denotation matrix of the final 50 breeding cycles of the experiment. An entropy value of 6 would indicate a completely random matrix. A value of 3 indicates a perfectly structured matrix, with one symbol per situation.

Table 2 shows the entropy of the denotation matrices over the last 50 breeding cycles of the experimental runs. Again, MacLennan and Burghardt's figures are taken directly from [16], and our own figures summarise 20 different runs. In the C^-L^- condition we found significantly *more* structure to the denotation matrices than did MacLennan and Burghardt, and in the C^+L^+ condition we found significantly *less*. Instead of the lowest entropy being associated with C^+L^+, we find it to be associated with C^+L^-. In other words, the most structured communication conventions develop in the communication only condition, and the addition of the learning rule only reduces that structure.

The differences between our findings and those of MacLennan and Burghardt should not be exaggerated. In all measurements, across all conditions, our figure was well within an order of magnitude of MacLennan and Burghardt's figure. Our interpretation of their experimental method may not reflect *exactly* their actual procedure, but at this point in time we have been unable to locate the source of the discrepancy. MacLennan and Burghardt's central result was successfully replicated: that communication, when enabled, leads to relatively high rates of fitness increase, and to the evolution of a structured "language" as evidenced by the denotation matrix.

4 Extension and critique

Having described the methods used by MacLennan and Burghardt, and noted the degree to which our results match theirs, we now wish to comment critically on certain aspects of their experiment. Several questions are raised as to what might be an appropriate methodology for studying the evolution of communication, and we hope to answer these questions in section 5.

	Mean	*SD*	Effect
Fitness increase			
C^-L^-	0.94	1.52	−4.5%
C^+L^-	18.6	7.05	+27.4%
C^+L^+	33.7	13.8	+218%
Final mean fitness			
C^-L^-	6.76	0.53	+0.23%
C^+L^-	14.47	2.83	+13.9%
C^+L^+	22.24	5.21	−51.8%

Table 3: Effect of random-order updating. Rate of fitness increase $\times 10^{-4}$ breeding cycles (determined by linear regression), and final mean fitness scores are shown, with means and standard deviations across 20 runs. The "effect" column compares the random-order results with our standard updating results (see table 1); note that if the updating method was not influencing the results, we would expect this value to be close to zero.

4.1 No geometry?

MacLennan and Burghardt claim that there are "no geometrical relations" [16, p. 166] among the simorgs. This is in keeping with their goal of constructing a synthetic world that is as simple as possible while still permitting communication to evolve. If the simorgs were arranged on a toroidal grid and could communicate only locally, for example, this would certainly complicate things.

However, in the current set-up, the simorgs are effectively arranged in a ring. As MacLennan and Burghardt [16, p. 170] put it, "The simorgs react one at a time in a fixed order determined by their position in a table." Thus there is at least a topology, if not a geometry: simorgs will tend to receive signals from their immediate neighbours in one direction, and send signals to their neighbours in the other direction.

The experiment could have been performed without this modest topological assumption if the simorgs were updated in a different random order at each time step. We modified our version of the program to use just such an updating procedure. Table 3 shows the rates of fitness increase and final fitness scores under this method.

There is a dramatic difference between the two updating methods. In the communication only (C^+L^-) and no communication (C^-L^-) conditions, similar performance is observed under both updating methods. The effect of the learning rule, on the other hand, depends very much on the updating method used: under random-order updating, the rate of fitness increase is much higher. Curiously, the rates of fitness increase under random-order updating come closer to the rates observed by MacLennan and Burghardt — perhaps this is a clue as to the cause of our differing findings.

Furthermore, random-order updating clears up an irksome feature of MacLennan and Burghardt's results. Fitness in the learning condition commences close to the random level of 6.25 (see the notes to table 1), which makes mean fitness directly comparable with the other conditions — note the 51.8% drop in final fitness scores. Under standard updating, a simorg will often have its look-up table corrected on the first time step of an environment cycle, then find itself in exactly the same context on the next four time steps, and score up to four "free hits". When simorgs are responding in a different random order each time step, it is no longer the case that a simorg will be communicating with the same near neighbours every time, and the learning rule loses this bonus property.

The most important point about the random updating procedure, however, is that it demonstrates that MacLennan and Burghardt's results could be dependent upon such apparently minor assumptions built in to their procedure. Their goal is to uncover general laws that can be translated back into the realm of real biology, but if the effect of learning on the evolution of communication is dependent on the updating method used, it is difficult to know what biological conclusions should be drawn. Does learning facilitate the development of a communicative system, or doesn't it?

4.2 Dialects or sub-optimal look-up tables?

MacLennan and Burghardt, noting the ambiguous symbol use evident in the denotation matrices, comment that "we cannot tell from [the denotation matrix] whether this multiple use of symbols results from two subpopulations or from individual simorgs using the symbol to denote two situations." [16, p. 179]. The idea that there could be subpopulations using different dialects seems quite plausible, especially given that the topology of the simorgs' environment (see section 4.1) ensures that simorgs will only be communicating with near neighbours. One can imagine a series of simorgs using variant A in one section of the ring, shading gradually into variant B in the opposite section, and back again.

MacLennan and Burghardt claim that the facts of the matter could easily be uncovered: given that the underlying finite state machines are available in computer memory, "there need be no mystery about how the simorgs are communicating, because the process is completely transparent." [16, p. 179]. However, they make no clear statement as to whether they in fact believe there are two or more subpopulations using variants of the evolved "language". MacLennan, in his earlier paper, is less conservative: "the differing use of symbols in various contexts makes it quite possible for every simorg to be using a different dialect of the 'language' manifest in the denotation matrix." [15, p. 653].

In an attempt to resolve this question, we used a convergence statistic in our experiments. We examined each

position on the genome in turn, and calculated the mean percentage of identical entries across the population of simorgs. Thus, a convergence statistic of 100% would indicate a population of simorgs with identical genomes and, thus, identical FSMs.

In runs of 5000 breeding cycles duration, the final convergence statistic was typically between 75% and 85%. This is not conclusive: it means that up to 25% of the simorgs could have been different from the norm, or that 25% of the genetic material of each simorg could be unique, and so leaves plenty of room for the possibility of different dialects. However, when the runs were extended to 2×10^4 breeding cycles or more, final convergence statistics in the C^+L^- condition were approximately 99.5%, and denotation matrices were qualitatively similar, i.e. they still showed ambiguous communication. It is implausible to suggest that there might be different dialects when the simorgs in a population are 99.5% identical to each other. We conclude that the suggestive ambiguity in the denotation matrices is nothing more than the net effect of (more or less) the whole population using a single, inefficient "language" that sometimes represents a state by more than one symbol, or uses one symbol to denote more than one state.

Despite their stated wariness about adopting any sort of intentional stance towards the simorgs [16, p. 163], we believe that MacLennan and Burghardt are not immune to the temptation to think of the simorgs as intentional agents, and that, in this case, that temptation has led them astray. Language without the scare quotes is undoubtedly the exclusive province of sophisticated intentional agents (argued in e.g. [1, 5]), but having drawn the analogy between human language and simorg communication, MacLennan and Burghardt were too ready to suspect that, like real language users, simorgs might have dialects. We are not claiming that they were wrong to draw the analogy in the first place; indeed, a central premise of adopting the intentional stance is to take such analogies very seriously. However, in dealing with simple systems like the simorgs, lower level mechanistic explanations are potentially open to us. The intentional story (dialects) can be shown, via the convergence statistic, to be inferior to the mechanistic story (less than ideal structure in the FSM look-up tables).

4.3 Consequences of the FSM look-up table approach

Imagine for a moment that you are a simorg. Disregarding the fact that simorg "decisions" are entirely determined by the look-up table, imagine that you have decided to emit a symbol. The only context that is important to you is the state of your local environment: you need to choose the right symbol to describe your situation, according to the "language" conventions that have developed. The identity of the symbol in the global environment is unimportant, because you're going to overwrite it anyway.

Similarly, if you're going to act, you don't care about the state of your local environment; you only want to interpret the global symbol in such a way as to correctly match the environment of the last emitter, and thereby score a point of fitness.

For the real simorgs of MacLennan and Burghardt, things are not this simple. There is no prior decision to emit or to act, only the consultation of a table with an entry for every possible combination of local and global environment states. As MacLennan and Burghardt put it, "finite-state machines have a rule for every possible condition." [16, p. 168].

Surprisingly, this means that the choice of the FSM architecture makes evolving a communication system harder for the simorgs than it might be under some other control architectures. For example, if during a particular run it became advantageous to reliably perform action-2 in response to symbol-7, FSM-controlled simorgs would have to ensure — through evolution or learning — that eight distinct entries in their look-up table came to be identical. That is, they would need to perform action-2 in response to symbol-7 in the context of eight different possible local environment states. By contrast, a simorg that was controlled by, for example, a classifier system (see e.g. [12, 24]) would need only to generate a single production rule: "perform action-2 in response to symbol-7". We have not yet investigated this empirically, but we offer as a hypothesis our suspicion that simorgs controlled by classifier systems would evolve significantly faster than FSM-controlled simorgs.

MacLennan and Burghardt did not believe that FSMs were the only architecture open to them, and adopted them for pragmatic reasons. However, if an arbitrary choice of control architecture is influencing their results in unexpected ways, it is again difficult to see how their conclusions can be reliably translated back to biology.

4.4 Counter-intuitive optimal strategy

The optimal strategy for the simorgs, at least at the population level, must be to act as often as possible, and to emit as infrequently as possible. This is because emitting scores no fitness points directly.

The best way for the simorgs to achieve this is to build up a link between a *single* global symbol γ and a *single* local state λ. A situation develops where simorgs always "blindly" respond with action-λ, unless they are in state λ themselves, in which case they emit γ. Imagine all the simorgs acting in this way: it is clear that they would no longer need to be concerned about the particular identity of the symbol in the global environment. They would *know* that it will always be a γ, and that it will always reliably indicate that the last emitter (whoever that may be) is in state λ.

Assuming 8 symbols and 8 local states, this means that a successful communication will take place 7/8 of the time[4]. This would translate as a mean fitness of 87.5 — a very high value relative to the results presented in tables 1 and 3. In general the maximum fitness will be equal to $100 \times \frac{L-1}{L}$, where L is the number of local states. To date we have only seen this phenomenon evolve spontaneously when $L \leq G \leq 4$, but the principle remains.

The trouble with this result is that one presumably does not want to call it an evolved communication system or "language", even though the simorgs are ostensibly fitter than ever before. If the global environment is (almost) always in the same state, it is difficult to describe it as carrying any information. The simorgs in such a situation appear to be exploiting a loophole in the experimental design.

MacLennan and Burghardt were aware of this possibility (see section 2.3). They saw it as most relevant to the C^-L^- condition, in that it provided an explanation for the otherwise mysterious increase in fitness observed. MacLennan [15, p. 653] felt that "in most cases [it] is a low level effect that is unintrusive and can be ignored".

4.5 Fewer symbols: faster improvement

The point outlined in 4.4 has a number of implications. Given that the optimal strategy involves the utilisation of only one symbol, we hypothesised that giving the simorgs progressively *fewer* symbols to work with would steer them towards that strategy and thus *improve* their performance. This contrasts with the intuitive hypothesis that n local states will require simorgs to use n symbols to denote them. MacLennan and Burghardt seem to have assumed the truth of the intuitive hypothesis: they speak of the ideal denotation matrix as having one symbol to denote each situation, and refer to the fact that $L = G$ as meaning that "there were just enough possible sounds to describe the possible situations." [16, p. 175].

To test our hypothesis we used the C^+L^- condition, held the number of local environment states constant at $L = 8$, and varied the number of global environment states G, i.e. the number of possible symbols, from eight down to one. For each case, 20 runs of length 5000 breeding cycles were conducted. The results are shown in figure 1.

Overall, higher rates of fitness increase were associated with smaller numbers of symbols. These results certainly contradict the intuitive view. In an effort to make a connection with real-world biology, one might argue that we are demonstrating the principle, described by Wiley [23], that small signal repertoires enhance the detectability of ritualised signals. The argument would be without merit, however. Wiley discusses the value of a small signal repertoire as a means of enhancing detection *in a noisy environment.* The simorgs' environment has no noise, and their perception of symbols is direct, immediate and reliable. Again, we claim that there is no easy biological translation of this observation concerning MacLennan and Burghardt's synthetic world.

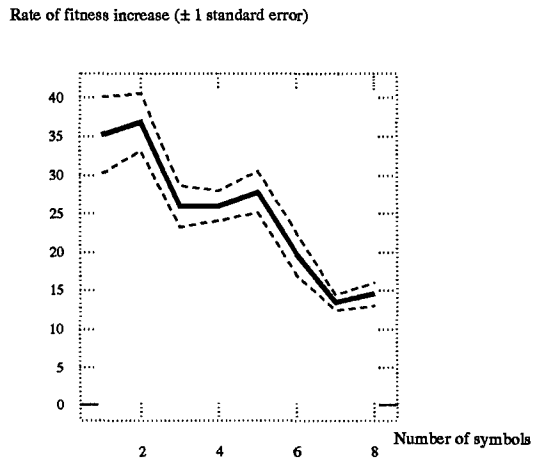


Figure 1: Mean effect on rate of fitness increase of varying G, the number of global symbols, while $L = 8$. Rate of fitness increase measured in units $\times 10^{-4}$ breeding cycles (determined by linear regression).

4.6 Symbol use over time

If the simorgs were evolving a "language", with an eventual one-to-one correspondence between global symbols and local states, we should observe a fairly even distribution of the G symbols. That is, simorgs should use each symbol about equally often. This follows from the fact that the distribution of local states is random and therefore uniform.

This is not observed, however. Popular symbols tend to get more popular; in the 20 runs of the C^+L^- condition reported in table 1, the mean usage of the most popular symbol at the end of the run was 41.45%. When longer runs were conducted, the popularity of the most popular symbol was even higher. Infrequently used symbols often dropped out of use altogether. This seems to indicate that the simorgs are drifting towards the optimal strategy described in 4.4.

5 Communication reconsidered

MacLennan and Burghardt clearly succeed in establishing that communication can occur in a particular sim-

[4]Discounting for a moment the unfortunate simorg who acts in response to an out of date symbol immediately after the local environments have been randomly reset; in the "experiential world" of the simorgs, this is an infrequent event.

ulated environment. They express the hope that their work will suggest general laws or principles concerning animal communication, but they are aware that "if the synthetic world is too alien, we may doubt the applicability to our world of any observations made of the former." [16, p. 166]. In section 4, we have pointed out various ways in which their synthetic world is indeed alien. Regrettably, it *is* difficult to see how certain aspects of MacLennan and Burghardt's results could be translated into real-world biology. However, we are optimistic about the usefulness of evolutionary simulations generally, and we would argue that the central problem with their work is simply that it tries to do too much, too soon.

There is not an established body of literature on simulating the evolution of communication. As things stand, communication is just one of the many biological phenomena that have come under attention from those pursuing the SAB/AL programme. We think it is safe to say that there are not yet any agreed-upon methods or landmark findings; there is only the central SAB/AL premise that evolution can be captured in a computer program, and a resolution to use ideas from biology, ethology, behavioural ecology, and signalling theory. In this context we believe that, despite its apparently simple nature, MacLennan and Burghardt's work is overly complex and ambitious.

MacLennan and Burghardt are trying to do a number of things at once. Primarily, they are attempting to provide an existence proof for the synthetic evolution of communication, and they make no secret of having constructed the synthetic world so that the simorgs will be likely to reproduce only if they cooperate (i.e. communicate) in the specified way. They are also examining a process by which arbitrary symbols can evolve to denote something in a simple "language". As they put it, "beyond merely detecting the presence of communication, we are also interested in studying its structure." [16, p. 173]. Further, because the simorgs must come to know not only the correlations between symbols and local states, but also when to act and when to emit, MacLennan and Burghardt are effectively looking at the evolution of turn-taking. Finally, they are interested in the effect of learning on the evolution of communication.

With the possible exception of the basic existence proof, each of these phenomena are poorly understood, and each is worthy of a separate, narrowly-focused simulation experiment. When all of these questions of interest are thrown in together, they interfere with each other and make the extraction of general principles impossible. For instance, in trying to push the simulation towards communication, they choose to reward both the sender and the receiver of a message, and in an effort to leave things open-ended enough for spontaneous symbol-meanings to develop, they use the FSM architecture. But what is the relative importance of these factors in causing the observed results? MacLennan and Burghardt allow spontaneous strategies for emitting vs. acting to develop amongst the simorgs, presumably to leave them as unconstrained as possible, but this decision creates the loophole described in section 4.4. Would the same type of communication develop if the simorgs were constrained to be senders and then receivers in turn?

In principle, it may be that communication between simorgs is entirely dependent on their internal architecture, or on the fitness reward structure used, or some other quirk of the methodology — MacLennan and Burghardt themselves note that when the method for selecting parents was deterministic rather than stochastic, communication did not develop. It is not possible, from MacLennan and Burghardt's results alone, to determine any necessary or sufficient conditions for the evolution of communication; they are doing the equivalent of commencing the study of gravitation with a four- or five-body problem.

Of course, we are not claiming that if only the various factors bearing upon the behaviour of MacLennan and Burghardt's simorgs could be isolated, then the general principles governing naturally evolved communication would be laid bare. It is quite likely that there are complicated, non-linear interactions even in their small system. However, if we do not understand the effect of each factor alone (e.g. cost or benefit of communication, updating method, simorg architecture) then it would seem optimistic to hope to understand the complex case.

The difficulties with MacLennan and Burghardt's experiment can be seen in another light: they compare synthetic ethology favourably with empirical ethology in that experiments in the former are repeatable, and full access to all variables is possible. However, this comes at a price. MacLennan and Burghardt are forced to rigorously specify the environment and the internal nature of the simorgs, making several ad hoc decisions along the way. In a sense, they have to go down to the level of simorg genetics. This is interesting, because one of the great strengths of ethology comes from what Grafen calls the "phenotypic gambit" [8, p. 6], in which genetics is almost entirely abstracted away, and broad behavioural strategies are considered at a functional level [5]. Most of the time, the conclusions so derived are borne out in the real world. The parallel to be drawn with MacLennan and Burghardt's experiment is that there is much to be done, using simulation methods, that does not buy into the question of internal architectures, but looks at one phenotypic characteristic at a time and assesses its effect on the evolution of communication. For example, one

[5] Grafen was specifically discussing behavioural ecology, the offshoot of ethology that concentrates on Tinbergen's third question: "What is this behaviour *for*?"

could simulate a population of agents who were either communicators or mutes, and then allow that population to evolve under different cost and benefit regimes for communicative behaviour. We might expect that when both the sending and the receiving agent benefit from communicative behaviour, then communicators will come to dominate the population. But what about when only the receiver benefits, or when the sender's benefit is relatively small? What happens when communicators will only signal to other communicators? This sort of simulation, taking up where the mathematical arguments of biologists such as Hamilton [9] and Grafen [7] leave off, would give us a sound basis for further investigations.

The best philosophical strategy for such future work is to adopt the intentional stance with respect to simulated organisms. Despite the fact that MacLennan and Burghardt at one point go too far, in ascribing high-level intentional phenomena such as language dialects to the simorgs (see section 4.2), we agree with Dennett [5, p. 265] about intentional accounts: "...in a nutshell, *they work*. Not always, but gratifyingly often." We are also confident that mechanistic explanations can peacefully co-exist with intentional ones; in the very simple simulations we are initially proposing, no doubt mechanistic accounts will predominate, with the balance gradually shifting as real-world complexity is incrementally approached.

MacLennan and Burghardt are at pains to *avoid* intentional talk when they define communication, and MacLennan [15] criticises denotational (i.e. intentional) theories of meaning. Nevertheless they rely on an analogy featuring rational, intentional agents — the story of the hunters — to make sense of their simulation, and they use denotation matrices to index the meaning of symbols: an intentional technique, in the sense of "aboutness", if ever there was one. We contend that to seek a non-intentional account of communication is to seek an oxymoron.

References

[1] J. Bennett. *Linguistic Behaviour.* Cambridge University Press, 1976.

[2] V. Braitenberg. *Vehicles: Experiments in Synthetic Psychology.* MIT Press, Cambridge, MA, 1984.

[3] G.M. Burghardt. Defining 'communication'. In J.W. Johnston, Jr., D.G. Moulton, and A. Turk, editors, *Communication by Chemical Signals.* Appleton-Century-Crofts, New York, 1970.

[4] P. de Bourcier and M. Wheeler. Signalling and territorial aggression: An investigation by means of synthetic behavioural ecology. In D. Cliff, P. Husbands, J.-A. Meyer, and S.W. Wilson, editors, *From Animals to Animats 3: Proc Third Int Conf Simulation of Adaptive Behavior*, Cambridge, MA, 1994. Bradford / MIT Press.

[5] D.C. Dennett. *The Intentional Stance.* Bradford / MIT Press, Cambridge, MA, 1987.

[6] S.J. Gould and R. C. Lewontin. The spandrels of San Marco and the panglossian paradigm: A critique of the adaptationist programme. *Proc Roy Soc*, B 205:581–598, 1979.

[7] A. Grafen. Biological signals as handicaps. *J Theo Bio*, 144:517–546, 1990.

[8] A. Grafen. Modelling in behavioural ecology. In J.R. Krebs and N.B. Davies, editors, *Behavioural Ecology: An Evolutionary Approach*, chapter 1, pages 5–31. Blackwell, Oxford, third edition, 1991.

[9] W.D. Hamilton. The genetical evolution of social behaviour. *J Theo Bio*, 7:1–52, 1964.

[10] D.G.C. Harper. Communication. In J.R. Krebs and N.B. Davies, editors, *Behavioural Ecology: An Evolutionary Approach*, chapter 12, pages 374–397. Blackwell, Oxford, third edition, 1991.

[11] R. Harré. Vocabularies and theories. In R. Harré and V. Reynolds, editors, *The Meaning of Primate Signals*, pages 90–105. Cambridge University Press, 1984.

[12] J.H. Holland. *Adaptation in Natural and Artificial Systems.* University of Michigan Press, Ann Arbor, 1975.

[13] J.R. Krebs and R. Dawkins. Animal signals: Mind reading and manipulation. In J.R. Krebs and N.B. Davies, editors, *Behavioural Ecology: An Evolutionary Approach*, chapter 15, pages 380–402. Blackwell, Oxford, second edition, 1984.

[14] D. B. Lewis and D. M. Gower. *Biology of Communication.* Blackie, Glasgow, 1980.

[15] B.J. MacLennan. Synthetic ethology: An approach to the study of communication. In C. G. Langton, C. Taylor, J. D. Farmer, and S. Rasmussen, editors, *Artificial Life II.* Addison-Wesley, 1991.

[16] B.J. MacLennan and G.M. Burghardt. Synthetic ethology and the evolution of cooperative communication. *Adaptive Behavior*, 2(2):161–188, 1994.

[17] J. Maynard Smith. *Evolution and the Theory of Games.* Cambridge University Press, 1982.

[18] J. Maynard Smith. Adaptationism and satisficing. *Behav Brain Sci*, 6:370–371, 1983.

[19] G.F. Miller. Artificial life as theoretical biology: How to do real science with computer simulation. CSRP 378, School of Cognitive and Computing Sciences, University of Sussex, 1995.

[20] R. Seyfarth, D.L. Cheney, and P. Marler. Monkey responses to three different alarm calls: Evidence of predator classification and semantic communication. *Science*, 210:801–803, 1980.

[21] C.E. Shannon and W. Weaver. *The Mathematical Theory of Communication.* University of Illinois Press, Urbana, 1949.

[22] G.M. Werner and M.G. Dyer. Evolution of communication in artificial organisms. In C. G. Langton, C. Taylor, J. D. Farmer, and S. Rasmussen, editors, *Artificial Life II.* Addison-Wesley, 1991.

[23] R.H. Wiley. The evolution of communication: Information and manipulation. In T.R. Halliday and P.J.B. Slater, editors, *Communication*, pages 82–113. Blackwell, Oxford, 1983.

[24] S.W. Wilson. Classifier fitness based on accuracy. *Evolutionary Computation*, 3(2), 1995.

[25] A. Zahavi. Mate selection — a selection for a handicap. *J Theo Bio*, 53:205–214, 1975.

[26] A. Zahavi. The cost of honesty (further remarks on the handicap principle). *J Theo Bio*, 67:603–605, 1977.

Collective Behavior by Modular Reinforcement-Learning Animats

Norihiko ONO Kenji FUKUMOTO Osamu IKEDA
Department of Information Science and Intelligent Systems
Faculty of Engineering, University of Tokushima
2-1 Minami-Josanjima, Tokushima 770, Japan
Phone: +81 886 56 7509 FAX: +81 886 55 4424
Email: ono@is.tokushima-u.ac.jp

Abstract

Several attempts have been reported to let multiple monolithic reinforcement-learning agents synthesize coordinated decision policies needed to accomplish their common goals effectively. Most of these straightforward reinforcement-learning approaches, however, scale poorly to more complex multi-agent learning problems, because the state space for each learning agent grows exponentially in the number of its partner agents engaged in the joint task. In this paper, we consider the Pursuit Problem as a multi-agent learning problem which is computationally intractable by these straightforward approaches. We show how successfully a collection of modular Q-learning hunter agents synthesize coordinated decision policies needed to capture a randomly-fleeing prey agent effectively, by specializing their individual functionality and organizing herding behavior.

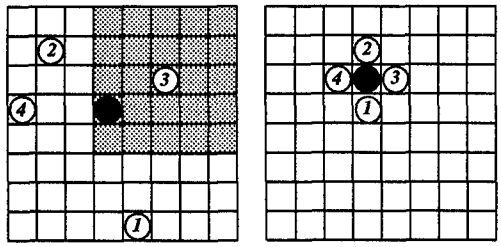
1 Introduction

In attempting to let artificial organisms or simple reactive robots synthesize some coordinated behavior, several researchers in the fields of artificial life and robotics have applied monolithic reinforcement-learning algorithms to multi-agent learning problems(e.g., [3, 7, 8, 9, 11, 12, 13]). In most of these applications, only a small number of learning agents are engaged in their joint tasks and accordingly the state space for each agent is relatively small. This is the reason why monolithic reinforcement-learning algorithms have been successfully applied to these multi-agent learning problems. However, these straightforward applications of reinforcement-learning algorithms do not successfully scale up to more complex multi-agent learning problems, where not a few learning agents are engaged in some coordinated tasks[12]. In such a multi-agent problem domain, agents should appropriately behave according to not only sensory information produced by the physical environment itself but also that produced by other agents, and hence the state space for each reinforcement-learning agent grows exponentially in the number of agents operating in the same environment. Even simple multi-agent learning problems are computationally intractable by the monolithic reinforcement-learning approaches. Considering the Pursuit Problem[1] as such a multi-agent learning problem, we show how successfully modular Q-learning prey-pursuing agents synthesize coordinated decision policies needed to capture a randomly-moving prey agent, by taking advantage of collective behavior.

2 The Pursuit Problem

In this paper, we treat the Pursuit Problem originally suggested in [1]. In an $n \times n$ toroidal grid world, a single *prey* and four *hunter* agents are placed at random positions in the grid, as shown in Fig.1 (a). The hunters operate attempting to capture the randomly-fleeing prey. At every time step, the prey and each hunter select their own actions independently without communicating with each other and accordingly perform them. Each agent has a repertoire of five actions. It can move in one of four principle directions (north, east, south, or west), or alternatively remain at the current position. The prey can not occupy the same position that a hunter does. However, more than one hunter can occupy the same position. So the prey can not move to a position which has been occupied by a hunter, and vice versa. A hunter has a limited visual field of depth d ($2 \times d + 1 \leq n$), and can accurately, locates the relative position and recognizes the type(*hunter* or *prey*) of any other agent operating within its visual field, and it determines its next action according to the perceived information[1]. The prey is captured, when all of its four neighbor positions are oc-

[1] In our preliminary attempt[10], we assumed that a hunter accurately locates the relative position and recognizes the identifier of any other agent operating within its visual field(each agent has a unique identifier), and we employed yet another modular Q-learning approach to the problem.



(a) An Initial State (b) A Goal State

Figure 1: The Pursuit Problem. A hunter and a prey are indicated by a white and black circles, respectively. With each hunter an identifier (1,2,3, or 4) is associated, but it is invisible to other hunters. A hunter is able to accurately recognize the relative position of any other agent in its visual field of depth d. When $d = 2$, the hunter 3's visual field is indicated by a shaded square in (a). The relative position of the prey to the hunter 3 is $(-2, -1)$ in this initial configuration.

cupied by the hunters, as shown in Fig.1 (b). Then all of the prey and hunter agents move to new random positions in the grid world and the next trial starts. The ultimate goal for the hunters is to capture the prey as frequently as possible.

This problem has been treated as a canonical problem in the field of distributed artificial intelligence (DAI), but never as a multi-agent reinforcement-learning problem. Researchers in DAI have attempted to suggest some coordinated strategies, based on their proposed distributed problem solving principles, and have evaluated the performance(e.g., [4, 5]). In this paper, we attempt to let reinforcement-learning hunter agents spontaneously synthesize some collective strategies. As a basic learning algorithm, we have determined to employ Q-learning. However, the straightforward application of monolithic Q-learning to this kind of multi-agent learning problems brings about the problem of combinatorial explosion in the state space. Since multiple learning agents, each observing its partners' behavior, have to jointly perform a coordinated task, the state space for each learning agent grows exponentially in the number of its partners. For example, let us represent the state of a hunter agent by a combination of the relative positions of other agents, where the positions are represented by some unique symbols when they are not located within the hunter's visual field. Although there are only a small number of collaborative agents involved, the size of state space for a hunter equals $4m^2 + 6(m^2-1)^2 + 2(m^2-1)(m^2-2)^2 + (m^2-1)(m^2-2)(m^2-3)^2/6$ where $m = 1+2\times d$; the four terms in this expression stand for the cases when a total of 1, 2, 3, or 4 positions in the hunter's visual field are occupied by hunters[2]. The state space size is enormous even when the visual field depth d is small; e.g., at $d = 3$ and $d = 4$, it amounts to $1,021,700$ and $7,445,764$, respectively.

For this reason, we have decided to construct each hunter agent by a variant of Whitehead's modular Q-learning architecture[16] as shown in Fig.2. This architecture consists of three learning-modules and a single mediator-module. Each learning-module focuses on specific attributes of the current state and performs Q-learning. More specifically, at each time step, a hunter agent scans in a specific order individual positions within its visual field. Its i-th learning-module L_i receives only the relative position of the prey and that of the i-th partner that have been located during the scan, and ignores any information concerning the other partners. Its state is represented by a combination of these relative positions and its size equals $m^4 + 1$. This size of state space is tractable by the monolithic Q-learning algorithm when d is relatively small. On the other hand, a mediator-module, without learning, combines learning-modules' decision policies using a simple heuristic decision procedure and determines a final decision by the corresponding agent. In our preliminary experiments, the mediator-module selects the following action:

$$arg\max_{a\in A} \sum_{i=1}^{3} Q_i(x_i, a)$$

where Q_i denotes the action-value function sustained by the learning-module L_i. This corresponds to what is called *greatest mass* merging strategy[16]. Intuitively, this is a selection by majority among actions proposed by individual learning-modules.

Our modular Q-learning architecture is based on the original one proposed in [16], but it is not identical. The architecture in [16] was particularly designed for constructing a *single* reactive planner pursuing a fixed set of dynamically-activated multiple goals. It was primarily employed for shrinking the planner's enormous state space caused by the existence of multiple goals. Here, our architecture is designed for shrinking each agent's intractably enormous state space caused by the existence of its partners jointly working in the same environment. Note also that all the learning-modules, in our modular architecture, are always active and participate in the reinforcement-learning task unlike in the Whitehead case.

3 Results

So far as we know, no one has attempted to let hunter agents in the pursuit problem spontaneously synthesize a coordinated behavior and our modular Q-learning

[2]Note that a hunter and the prey are not allowed to occupy the same position at a time and at least one position within a hunter's visual filed is occupied by the hunter itself.

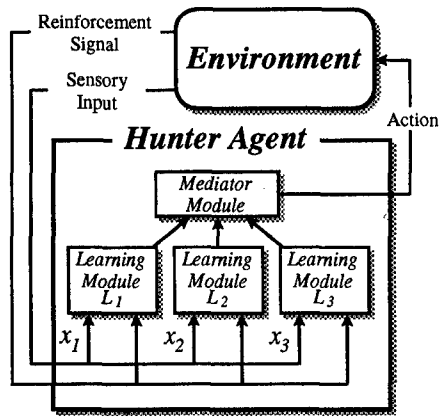


Figure 2: Modular Architecture for a hunter agent. At each time step, a hunter agent scans in a specific order individual positions within its visual field. Its constituent learning-module L_i only recognizes the prey and i-th partner that have been located during the scan, and does not concern any other partners' positions.

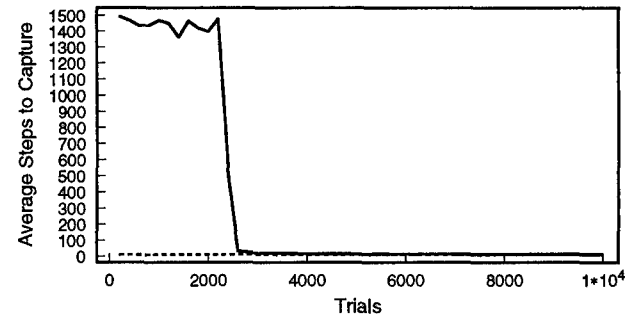
hunters have not been guaranteed to synthesize any of such behavior. So the depth of each hunter's visual field was set to a small value of 3. By changing the dimension of the grid world, we observed how our learning hunter agents can reduce the average number of time steps needed to capture the prey.

Our experiment consists of a series of trials, each of which begins with a single prey and four hunter agents placed at random positions and ends when the prey is captured. A trial is aborted when the prey is not captured within 1,500 time steps[3]. Upon capturing the prey, individual hunters immediately receive a reward 1.0, and accordingly all of its constituent learning-modules uniformly receive the same reward regardless of what actions they have just proposed. Our hunters receive no reward in any other case. The learning rate α and discounting factor γ are set to 0.1 and 0.9, respectively. Initial Q-values for individual state-action pairs are randomly selected from an interval $[0.01, 0.1]$.

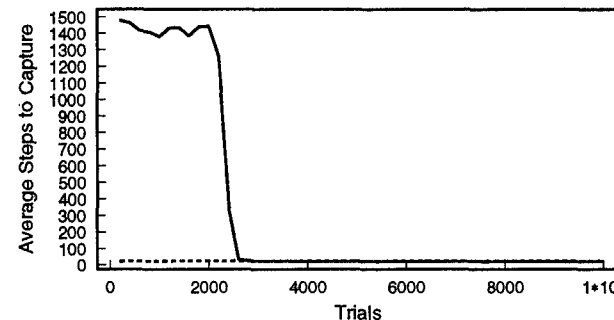
The typical results of our experiments are shown in Fig.3. Solid lines in this figure indicate the performance by the modular Q-learning hunters. At initial trials, the hunters can not capture the prey effectively, but shortly they start improving their performance dramatically. Eventually they come to steadily accomplish their task at every trial in any runs of our experiments. The meanings of dashed lines will be explained later.

To see the efficacy of our modular Q-learning over a straightforward monolithic version, we tried to solve the pursuit problem using a monolithic Q-learning approach.

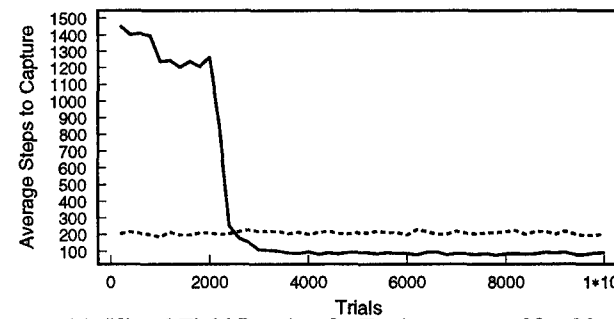
[3]Without aborting any trials, our learning hunters get the results which are comparable to those shown in this section.



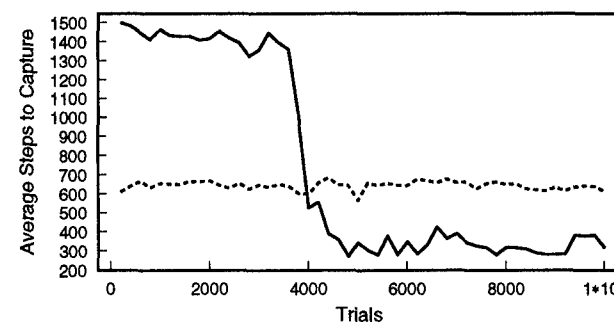
(a) Visual Field Depth = 3, Environment = 7 x 7



(b) Visual Field Depth = 3, Environment = 10 x 10



(c) Visual Field Depth = 3, Environment = 20 x 20



(d) Visual Field Depth = 3, Environment = 30 x 30

Figure 3: Performance of Modular Q-learning Hunter Agents. Four hunters, each with a fixed visual field depth 3, are operating together in the environments with various dimensions. Solid lines indicate the performance by modular Q-learning hunters, while dashed lines the performance by manually-programmed hunters, which as explained later, attempt to independently locate the prey by moving randomly and then behave in a probabilistically optimal way as soon as locating it.

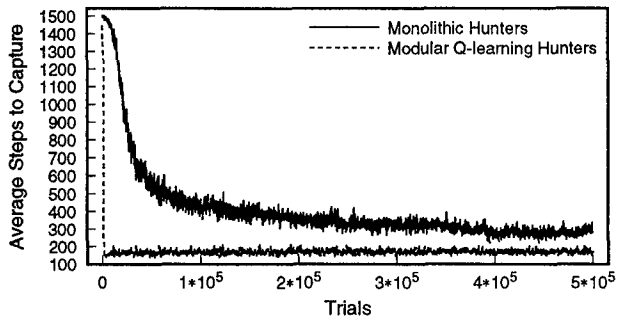


Figure 4: Performance of Monolithic Q-learning Hunter Agents. A typical performance by monolithic Q-learning hunters is shown as well as that by the modular Q-learning ones. In either case, each hunter's visual field depth equals 2 and a 20×20 environment is employed. The same learning rate, discounting factor, and Q-value initialization scheme as those described above are employed here.

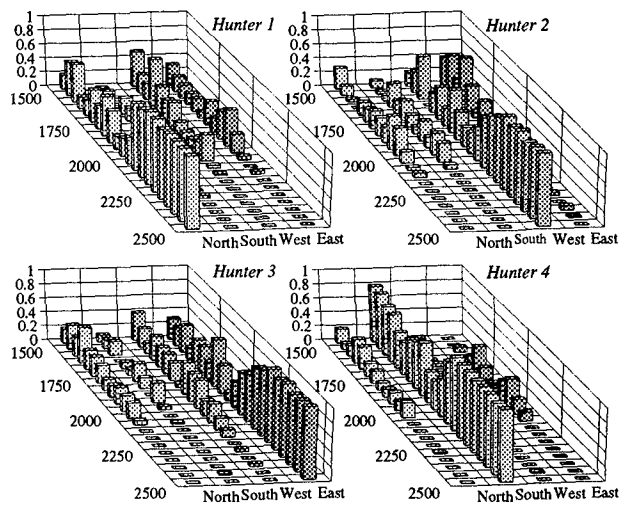


Figure 5: Emergence of Functionality-Specialization.

However, it turned out difficult to implement each hunter by a monolithic Q-learning even when each hunter's visual field depth is small, say 3. So we set the visual field depth to 2 in order to show the monolithic Q-learning hunters' learning performance. In Fig.4, the performance by the implemented monolithic Q-learning hunters and by the modular Q-learning hunters are shown. It may be possible to improve the monolithic Q-learning hunters' performance by fine tuning of the learning parameters, but limited computational experience showed that any dramatic improvement is difficult.

4 Synthesized Collective Behavior

Our modular Q-learning hunter agents exhibited two kinds of biologically-interesting collective behavior, namely, *herding* and *functionality-specialization*. During the course of dramatically improving their performance, individual hunters are specializing their functionality. To capture the prey, individual hunters have to exclusively occupy one of four positions surrounding it. Let us call these positions *responsible positions* of corresponding hunters. Eventually, our modular Q-learning hunters attempt to improve their pursuit performance by completely fixing their individual responsible positions, i.e., by specializing their functionality. Fig.5, corresponding to a result shown in Fig.3(c), illustrates how such a functionality-specialization is typically developed among hunters. Each of these four 3-D graphs shows a transition of the probability distribution over responsible positions (indicated by relative directions to the prey) taken by the corresponding hunter. At trials 1,500–2,000, hunters have not fixed their own responsible positions, but after trial 2,200, they have determined their own responsible positions and during this change in their behavioral patterns, their overall performance was dramatically improved.

Besides specializing their functionality, our hunters obtain a kind of herding behavior. Even if establishing the above-mentioned static role-assignment, hunters can not always accomplish the capturing task effectively until they have obtained an effective way of collectively locating the prey, since each of them can only observe the limited portions of the grid world. To see this, we manually-programmed a set of four hunters. We assigned individually these hunters' responsible positions in advance, and then let them (i) attempt to independently locate the prey by moving randomly and (ii) reach their responsible positions in a probabilistically optimal way as soon as locating it. The dashed lines in Fig.3 show typical performance by thus programmed hunters. Note that the larger environment, the more clearly our learning hunters outperform these manually-programmed hunters.

Our learning hunters, on the other hand, do not attempt to locate the prey independently, because it is not an effective way though the prey behaves randomly. Instead they locate it in a coordinated manner. Once having located a partner, a hunter attempts to keep the partner in sight so long as the prey has not been located and thereby it attempts to make a herd together with the partner. By herding in this way, hunters can operate as each other's eyes and compensate their limited perceptual capabilities. However, their behavior suddenly changes once one of herding members has happened to locate the prey. The member quickly attempts to reach its responsible position and the other ones follow it and

accordingly are likely to locate the prey. When all of hunters have located the prey, they can easily capture it.

Typically, the prey capturing process by our modular Q-learning hunters consists of two phases: (i) a *collective search phase* where all of hunters attempt to identify the prey's position by configuring a herd, and (ii) a *surrounding phase* where each hunter having identified the prey's position attempts to reach its fixed responsible position in an effective way.

To illustrate their collective behavior, a typical pursuit process by our learning hunters is shown in Fig.6. In this case, the environment is a 20×20 toroidal grid, and each hunter's visual field depth is set to 3. The hunters have experienced some $10,000$ trials. At this trial, a single herd has been developed at early time steps. By moving in a herd, the hunters attempt to collectively search the prey. Their behavior suddenly changes once one of them has happened to locate the prey, and eventually they capture it successfully.

5 Synthesis of Collective Behavior

Multi-agent reinforcement-learning is a difficult problem, especially when some forms of joint operation are needed for the agents. We do not say our modular reinforcement-learning approach be generally applicable to any multi-agent problem solving. To clarify this, we explain below how the learning solves this problem.

Although it might sound strange, our learning hunters acquire the above-mentioned herding behavior first and then specialize their individual functionality. Our hunters learn to make a herd even at initial trials, which depends on some attributes of the problem itself and our learning and state representation schemes.

One of the characteristics of the multi-agent reinforcement-learning problem, like the pursuit problem, is that learning agents can not receive meaningful or positive reward frequently and they are often forced to perform their initial learning without receiving any positive reward, particularly when engaged in some joint tasks. When operating under such a condition, Q-value for each state-action pair experienced by a modular Q-learning agent's learning-modules is repeatedly reduced. Note that the more often the same state-action pair has been experienced, the lower its corresponding Q-value. Thus, at initial trials, the modular Q-learning agents come to not select those actions that take them to frequently-experienced states.

The pursuit problem has similar characteristics. In fact, our hunters have to operate without receiving any positive reward at their initial trials. When its visual field is much smaller than the environment, a hunter often operates without identifying any other agents. Furthermore, any state of its learning-modules where a hunter can not identify any other agents is equally represented by a unique symbol. Note that the larger the

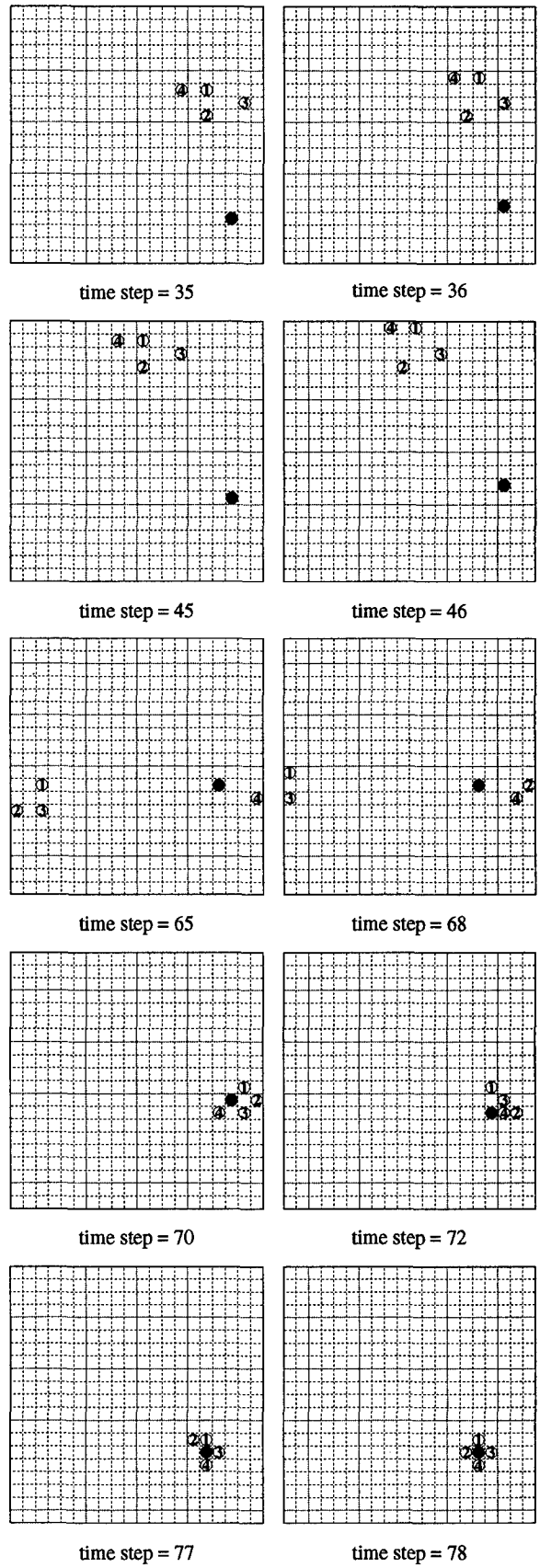


Figure 6: A Typical Pursuit Process.

environment compared with its visual field, the more often those states represented by the unique symbol are experienced in general. Thus, any state where no other agents are visible comes to seem less desirable for the hunter's learning-modules than the other states. This characteristics of the learning processes encourages the hunters to quite quickly acquire a trait to remain close to other agents, and accordingly get a good chance of capturing the prey. Without acquiring this trait and accordingly herding behavior, our learning hunters can not capture the prey effectively, because they have to occupy all of the four neighbor positions of the prey at a time.

Furthermore, the hunters' performance relies on our implementation details. It is sensitive to the order in which they scan positions within their individual visual fields. To point out this, we performed experiments using several scanning orders as shown in Fig.7. The corresponding results are shown in Fig.8. As shown in the figure, the scanning orders reflecting the Manhattan distance between a learning hunter and the other agents provide slightly better results than those reflecting the chessboard distance, when the same scanning directions, i.e. increasing or decreasing, are used, which may be due to the constraints posed on the hunters' repertoire of possible actions.

Note that when hunters scan within its visual field in a decreasing order, they fail to acquire effective policies. This is because the correspondence is subject to be dynamically destroyed between a learning-module and other agents identified by the module when such a scanning order is used. Suppose, for example, one partner remains in neighbor position to a learning hunter. Whenever another partner enters the hunter's visual field, the correspondence between the hunter's learning-modules and those partners being watched by the modules are changed.

46	38	26	14	27	39	47
40	28	15	6	16	29	41
30	17	7	2	8	18	31
19	9	3	1	4	10	20
32	21	11	5	12	22	33
42	34	23	13	24	35	43
48	44	36	25	37	45	49

(a) A Manhattan Addressing

26	27	28	29	30	31	32
33	10	11	12	13	14	34
35	15	2	3	4	16	36
37	17	5	1	6	18	38
39	19	7	8	9	20	40
41	21	22	23	24	25	42
43	44	45	46	47	48	49

(b) A Chessboard Addressing

Figure 7: Investigated Scanning Orders. Two independent addressing schemes of the positions within a hunter's visual field are shown. The visual field depth is supposed here to be 3. The addressing in (a) reflects the Manhattan distance of each position to the center of visual field, while the chessboard distance is used in (b). By scanning these positions in increasing and decreasing orders, four kinds of the scanning orders are defined and investigated.

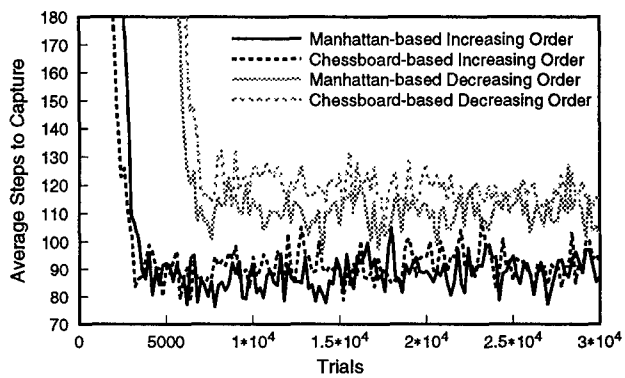


Figure 8: Learning Performance by Modular Q-learning Hunters Employing Different Scanning Orders.

6 Concluding Remarks

Recent attempts to let monolithic reinforcement-learning agents synthesize some of coordinated relationships among them scale poorly to more complicated multi-agent learning problems where multiple learning agents play different roles and work together for the accomplishment of their common goals. These learning agents have to receive and respond to various sensory information from their partners as well as that from the physical environment itself, and hence their state spaces are subject to grow exponentially in the number of the partners. As an illustrative problem suffered from this kind of combinatorial explosion, we have treated the pursuit problem, and shown how successfully a collection of modular Q-learning hunter agents synthesize coordinated decision policies needed to capture a randomly-fleeing prey agent effectively, by specializing their functionality and acquiring herding behavior.

Since the learning agents in this problem have a limited sensor area, even a single monolithic Q-learning agent is not guaranteed to learn optimal behavior. Multi-agent learning is a difficult problem in general, and the results we obtained strongly rely on specific attributes of the problem. But the results are quite encouraging and suggest that our modular reinforcement-learning approach is promising in studying adaptive behavior of multiple autonomous agents. In the next phase of this work we intend to consider the effects of the modular approach on other types of multi-agent learning problems.

There are many interesting questions regarding our work, e.g., relationships between the modular Q-learning approach and a number of generalization techniques on reducing state spaces (e.g., [2, 6]) which could be applied to multi-agent reinforcement-learning. All these remain for the future research.

References

[1] Benda, M., V.Jagannathan, and R.Dodhiawalla: On Optimal Cooperation of Knowledge Sources, Technical Report BCS-G2010-28, Boeing AI Center, 1985.

[2] Chapman, D. and L.P.Kaelbling: Input Generalization in Delayed Reinforcement Learning: an Algorithm and Performance Comparison, Proc. IJCAI-91, 1991.

[3] Drogoul, A., J.Ferber, B.Corbara, and D.Fresneau: A Behavioral Simulation Model for the Study of Emergent Social Structures, Toward a Practice of Autonomous Systems (Proc. the First European Conference on Artificial Life), The MIT Press, 1991.

[4] Gasser, L. *et al.*: Representing and Using Organizational Knowledge in Distributed AI Systems, L.Gasser, and M.N.Huhns (Eds.): Distributed Artificial Intelligence, Vol.II, Morgan Kaufmann Publishers, Inc., 1989.

[5] Levy, R., and J.S.Rosenschein: A Game Theoretic Approach to Distributed Artificial Intelligence, MAAMAW'94 Pre-Proc. the 3rd European Workshop on Modeling Autonomous Agents in a Multi-Agent World (available as technical document D-91-10 of German Research Center on AI), 1991.

[6] Munos, R., and J.Patinel: Reinforcement Learning with Dynamic Covering of State-action Space: Partitioning Q-learning, From Animals to Animats *3*, The MIT Press, 1994.

[7] Ono, N., and A.T.Rahmani: Self-Organization of Communication in Distributed Learning Classifier Systems, Proc. International Conference on Artificial Neural Nets and Genetic Algorithms, 1993.

[8] Ono, N., T.Ohira, and A.T.Rahmani: Emergent Organization of Interspecies Communication in Q-learning Artificial Organisms, in F.Móran *et al.*: (Eds.) Advances in Artificial Life (Proc. Third European Conference on Artificial Life), Springer-Verlag, 1995.

[9] Ono, N., T.Ohira, and A.T.Rahmani: Evolution of Intraspecies Communication in Q-Learning Artificial Organisms, Proc. Fouth Golden West Conference on Intelligent Systems, 1995.

[10] Ono, N., O.Ikeda, and A.T.Rahmani: Synthesis of Organizational Behavior by Modular Q-learning Agents, Proc. the 1995 IEEE/Nagoya University World Wisepersons Workshop (WWW'95) on Fuzzy Logic and Neural Networks/Evolutionary Computation, 1995.

[11] Rahmani, A.T., and N.Ono: Co-Evolution of Communication in Artificial Organisms, Proc. 12th International Workshop on Distributed Artificial Intelligence, 1993.

[12] Tan, M.: Multi-agent Reinforcement Learning: Independent vs. Cooperative Agents, Proc. 10th International Conference on Machine Learning, 1993.

[13] Yanco, H., and L.A.Stein: An Adaptive Communication Protocol for Cooperating Mobile Robots, From Animals to Animats *2*, MIT Press, 1992.

[14] Watkins, C.J.C.H.: Learning With Delayed Rewards, Ph.D.thesis, Cambridge University, 1989.

[15] Werner, G.M., and M.G.Dyer: BioLand: a Massively Parallel Simulation Environment for Evolving Distributed Forms of Intelligent Behavior, in H.Kitano (Eds.): Massively Parallel Artificial Intelligence, AAAI/MIT Press, 1994.

[16] Whitehead, S. *et al.*: Learning Multiple Goal Behavior via Task Decomposition and Dynamic Policy Merging, in J.H.Connell *et al.* (Eds.): Robot Learning, Kluwer Academic Press, 1993.

Robotic "Food" Chains: Externalization of State and Program for Minimal-Agent Foraging

Barry Brian Werger and Maja J Matarić
Interaction Laboratory
Department of Computer Science
Brandeis University
Waltham, MA 02254
barry@cs.brandeis.edu, maja@cs.brandeis.edu

Abstract

This paper describes experiments inspired by theoretical work on information invariants ([Donald 1995], [Donald et al 1994]), a means of comparison and a methodology for design of single- and multi-agent systems. Analysis reveals the environmental information that the systems assume and exploit, while the design methodology seeks to move information and processing into the "physical" environment and task mechanics. The approach raises the issue of agents actively recording information, or even "programs," into the physical environment. This paper provides an example system that dynamically encodes information and "programs" into its physical environment.

The second source of inspiration for this work is the natural phenomenon of ant pheromone trail formation, shown to involve agents with simple, local control that encode information into the environment to arrive at globally complex behavior. Analogously, our robotic system actively encodes information into its physical environment in order to reduce sensing, actuation, and computational requirements. Thus, "minimal" agents with local sensing and action form a system that dynamically and globally adapts to environmental changes. We discuss how moving information and "processing" into the shared physical environment improves our ability to generate complex global behaviors from simple locally interacting agents.

1 Introduction and Motivation

While current trends in robotics towards situated, embodied, multiple agents have provided numerous systems that react effectively and robustly to their environments, they have dealt only obliquely with the deliberate manipulation of the environment by the agents. Systems that implement behaviors such as aggregation, dispersion, and flocking [Matarić 1995] involve agents which, through their "physical" presence, influence the behavior of other agents in a manner that is more than mere "interference"; [Beckers et al 1994] describes a task where physical effects of task performance allow a simple, local control strategy to produce a consistent global behavior; and work in behavioral economics and "robot ecology" (e.g., [McFarland 1994], [Steels 1994]) has investigated the influence agents have on each other through the use and production of shared, limited resources.

We have been inspired by the elegant simplicity of natural forms of direct environmental modification such as territorial marking or pheromone trails. These phenomena exploit the benefits of having agents deliberately encode information into the physical environment. As discussed in [Aron et al in press], [Goss et al 1989], [Muller and Wehner 1988], and [Hölldobler 1990], the release of pheromones by ants leads to trails that can be differentiated by pheromone "strength," which is a function of frequency of use and decay. If pheromones are released only during certain phases of tasks (such as carrying some item back to the nest), then trails can begin to form *efficient* paths to useful locations, such as rich supply areas. This, combined with a very simple control strategy of probabilistically choosing the most frequently used path, leads to group behavior that adjusts to follow dynamically determined shortest paths to dynamically determined useful destinations.

The ability to take advantage of information "encoded" into the physical environment through task mechanics has recently been under investigation from the perspective of information invariants ([Donald 1995], [Donald et al 1994]), which seeks to examine the interaction between sensing, computation, communication, and task mechanics in the performance of distributed manipulation tasks. This approach has provided some theoretical basis for comparing sensori-computational systems, and some steps towards a methodology for design

of efficient distributed manipulation systems. Specifically, a number of systems are demonstrated which take advantage of physical effects of task dynamics to dramatically reduce the amount of sensing, computation, and communication which naively seems "necessary," and a methodology for minimalizing such requirements is proposed. However, work on this approach "is still biased towards sensing, and it remains to develop a framework that treats action and sensing on an equal footing" [Donald et al 1994].

Two questions raised by this research are: 1) the ability of agents to externalize, or encode "state" into the physical environment, and 2) the ability to do the same with "programs." We believe that the ant pheromone trails discussed above can be viewed as "state," and possibly even as "programs" physically encoded into the environment, and that a similar system can be employed by robots to create distributed physical representations - or even distributed physical "programs" - in their environment. In this paper, we present such a system of autonomous mobile robots that modifies its environment in way that allows dynamically changing, globally position-dependent tasks to be performed through local physical contact and very simple control rules. We discuss this system as both the object of analysis and inspiration for development of an extended information invariant-based approach we hope to develop in the future, including making steps towards extension of the methodology for the development of distributed manipulation protocols [Donald et al 1994].

2 The Foraging Task

Variations of *foraging* - collecting items from the environment and depositing them at a specific location - are examples of a common class of robotic tasks that requires some knowledge of global positioning for efficient performance. While purely stigmergic solutions have been found for tasks such as clustering items in the environment ([Beckers et al 1994]) and even sorting of scattered heterogeneous items into homogeneous clusters ([Deneubourg et al 1991]), tasks which require particular behaviors to take place at particular locations have so far relied upon some type of global position sensing, globally visible beacon, or random encounter of some locally-sensible position marker [1].

The following subsection gives an overview of the most commonly used sensory modalities and strategies for performing variations of the foraging task, and some of their associated requirements and overhead.

[1] With the notable exception of one of our inspirations, the simulated beacon chain system described in [Deneubourg et al 1990] and [Goss and Deneubourg 1991].

2.1 *Methods useful for single or multiple agents*

- *The Omniscient Planner:* The use of a planner that can "see" the whole environment and the forager's position within it, and plan accordingly. This is infeasible for non-trivial environments and group sizes.
- *Position/Orientation Sensing:* The use of absolute global position information. There are various ways to perform position and orientation sensing that can be considered to be effectively equivalent. Popular approaches include:

  Global Positioning System (GPS) and Compass: requires environmental preparation (the GPS), and a potentially sophisticated local sensor (the compass) that is typically very sensitive to environmental noise.

  Radio-Sonar Positioning System: triangulation based on time differences between arrival of sonar and radio signals provides position information. Heading information can be determined through analysis of change in position. This is the basis of several successful foraging systems ([Fontán and Matarić 1996], [Goldberg Matarić 1996], [Matarić 1995]), but requires preparation of the environment (radio-sonar broadcasters at precise locations), complicated sensing equipment (radios, sonar detectors), and triangulation computation.

  Dead-Reckoning: Determination of robot position and orientation through careful monitoring of actuator motion, such as wheel rotation. Does not require modification of the environment, but does require that initial location be known. This approach necessitates highly accurate and potentially complicated actuator motion sensing and calibration, and suffers from cumulative errors.
- *Taxis:* Following of some sort of beacon. This method involves some modification of the environment (the beacon), and is limited by the range of visibility.
- *Recognition of Unique Locations:* The use of local environmental features, through such means as vision or sonar, to identify certain locations (landmarks) to which the agent can orient itself. While this approach has been used successfully in various experiments (e.g., [Matarić 1992b], [Horswill 1993], [Gomi 1995]) it relies on having or acquiring some map representation of the environment, and sensing the landmarks sufficiently accurately to local-

ize within that map.

2.2 *Methods specific to multiple agents*

- *Pheromones:* Requires the ability to emit and detect the presence of varying concentrations of pheromones. Work towards a robotic odor sensing/depositing system has been done by [Russell 1995], but has not yet been applied to tasks.
- *Beacon Chains:* The same as taxis, except that the beacon does not have to be globally perceivable; instead, robots are equipped with beacons that can be left within visible range of each other, together forming chains of indefinite length. The approach requires the ability to distinguish between beacons, which must broadcast information regarding their distance (in beacons) from the home location.
- *Contact Chains:* Only simple local sensors (such as infrared or contact) are used in the process of chain formation and following. Agents follow a chain composed of the "bodies" of other agents towards the home location. We propose an extreme case in which the robots use a small number of the simplest, most reliable sensors available - contact sensors - as discussed below.

3 Robot Chains

The system we present involves the formation of "hand-holding chains" by a group of robots in order to provide local information sufficient for the performance of globally position-dependent tasks. The chain maintains contact with a starting point *Home*. Robots that are not currently part of the chain are able to follow the chain both away from *Home* and back towards it. The chains can adjust to link *Home* with other points, such as rich supply areas, and re-form when the supply diminishes or new deposits are discovered, and, potentially, be put into motion to completely sweep an area. Simple communication can be sent up and down the chain, allowing a wide range of fairly complex behaviors to emerge.

Earlier research on robot chains has been conducted through simulation in [Goss and Deneubourg 1991] and [Deneubourg et al 1990]. The chains were "line-of-sight" and required that each simulated robot be able to distinguish among beacons and locate those which communicated numbers representing distance (in beacon links) to *Home*. Unfortunately, this approach requires sophisticated sensors and transmitters, and given those, could still be sensitive to sensory errors and other noise in non-ideal environments.

To avoid these problems, the approach to chaining that we present uses only sensors that operate within the range of physical contact - microswitches and break-beams. The microswitches indicate physical contact with another object. Each member of the chain actively maintains contact with the link ahead and behind by touch, through microswitches.

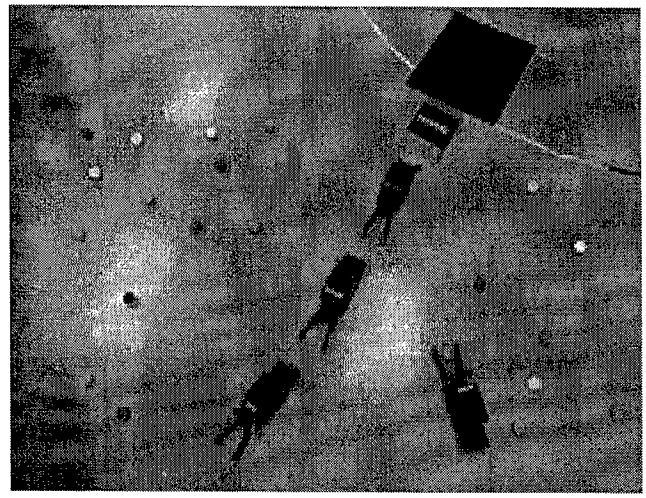


Figure 1: A robot returns to the chain with a puck after a circular excursion.

Limited communication is implemented through the same mechanisms to allow for chain maintenance. The most common type of communication is phatic, intended only to assert the existence of the line of communication (i.e., the integrity of the robot chain). This is implemented as a "double tap." One robot begins the communicative act (Figure 2) by moving enough to tap the robot ahead or behind twice and returning to its (approximate) initial position. The tapped robot answers by tapping back twice and returning. Two taps are used to distinguish communication from the many random taps of other robots in the environment.

More informative communication can be performed similarly, with contact held for a fixed period, or taps added, at points B, D, and F of Figure 2. Many interesting behaviors require no more than just this simple 1-bit phatic communication, but it is possible to pass more elaborate messages through combinations of "short" and "long" taps.

The basic behaviors involved in chain formation and maintenance are:

- *HomeLink:* Remains still, except to maintain communication with next link in the chain.
- *MiddleLink:* Maintains communication with "next" and "previous" links by returning taps and passing messages.
- *EndOfChain:* Maintains communication with link "ahead," assists in positioning of new links by interacting in the alignment process, and establishes communication with newly aligned

links to pass on the status of *EndOfChain* before becoming a *MiddleLink*. If not useful for a given period of time, it communicates its intention to the "previous" link to transfer *EndOfChain* status and leaves the chain.

- *JoinChain:* Works with *EndOfChain* to align a robot properly at the end of the chain, establish communication with the current *EndOfChain*, and become the new *EndOfChain*.

Robots not part of the chain can determine which way along the chain *Home* is, and follow the chain towards or away from it, using a physical feature that allows simple sensors to determine a very rough estimate of heading relative to another robot.

Behaviors involved in chain following are:

- *GoHome:* Determine direction *Home* and follow chain in that direction.
- *GoOut:* Determine direction away from *Home* and follow chain in that direction.

The foraging task our chain-making system performs involves the collection of metal pucks scattered either randomly or in clusters around the test environment. Two types of searching are used to locate and retrieve pucks:

- *Random Search:* Robots search for pucks throughout the environment, then locate the chain randomly once carrying a puck. This is often performed when the end of the chain is reached.
- *ExcursionSearch:* Robots follow chain, occasionally taking roughly circular journeys into the area next to the chain (Figure 4.)

Modifications of the basic behaviors discussed above allow for dynamic adjustment of the chain to various environmental factors. As mentioned earlier, in certain tasks it is desirable for the chain to connect a rich supply directly to *Home*. One way for the chain to move towards such a configuration is for the links to collect statistics on the number of times they are tapped on each side, and gradually shift towards the side that sees the most "traffic." We see this as somewhat analogous to the gradual buildup of pheromones on paths frequently used by ants; it should eventually lead to the same type of convergence on a shortest path to a highly useful destination ([Aron et al in press], [Matarić 1990], [Goss et al 1989]).

4 Current Implementation

We have implemented a foraging system which gathers metal pucks distributed around an area to the *Home* location using only physical contact-level sensing. The system is designed for the foraging team to begin in the

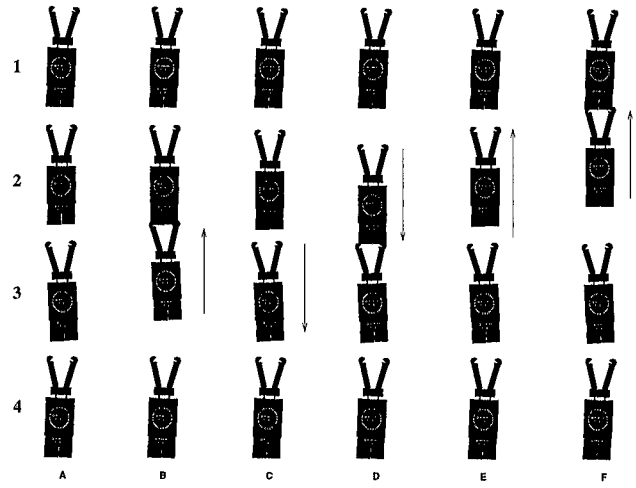


Figure 2: Communication passed down the chain. A) The chain in resting state. B) Robot 3 taps robot 2 twice to initiate the communication act C) 3 returns to normal position. D) 2 taps 3 twice to acknowledge communication. E) 2 returns to normal position, terminating communication act. F) 2 taps 1, initiating next communication act in the passing of the message down the chain. In non-phatic communication, stages B, D, and E are modified.

Home area. The system is functional with the following qualifications:

- *Sensing Home:* Infrared emitter/detectors with an effective range of less than 1 inch, located on the underside of the robots' fork arms, are used to determine when the robots are at *Home*, which is a nonreflective black area on the floor. This extremely short-ranged sensor can be replaced with a physical sensor of the same length capable of detecting some property of *Home*.
- *Initial Timing:* Currently the robots are powered up sequentially at appropriate times. In the future this will be done either by fixed timing based on unique ID numbers or, ideally, through messages passed back through the chain to the waiting team members.
- *Number of Robots:* As described below in 4.2, our herd of 20 robots is undergoing major renovations and modifications. The described experiments were performed with four robots fully capable of chain-building behavior, and two additional robots performing only behaviors based on following the chain. The length of our chains was thus limited to four, though we occasionally increased it by switching "dead" robots for chain links closer to *Home* (which are the least active) in order to re-use functional robots further down the chain.

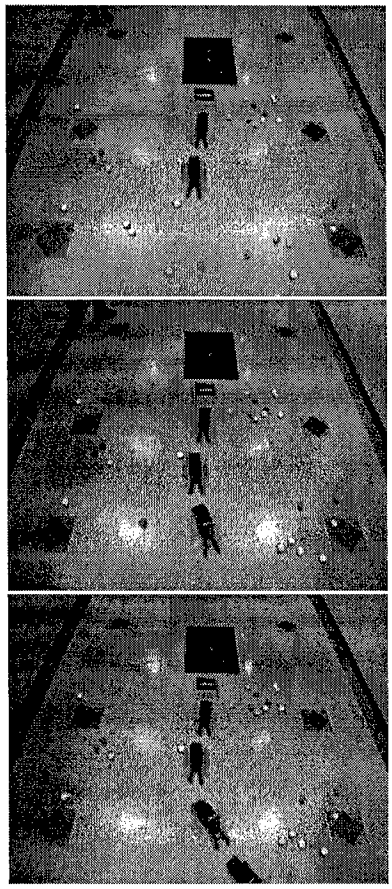


Figure 3: Three robots form a chain from *Home*

- *Environmental Assumption:* The current system assumes that the environment contains only robots, pucks, and *Home*.

4.1 Behaviors

4.1.1 Initial Chain Location - Skirt

Robots start gathered at *Home*. A behavior **Skirt** navigates to the edge of *Home*, then tacks along this edge until it encounters a physical obstacle projecting outside of the *Home* region. This obstacle is assumed to be the chain.

4.1.2 Chain Following - Tack

Chain following is performed through simple tacking. The following robot angles towards the chain until contact is made, backs off at a sharper angle, then angles back to make contact further down the chain. This tacking allows a following robot to round the end of the chain and continue down the other side. In current experiments we enforce directionality on chain traffic. Tacking is always done with the chain on the following robots' left side; thus the right side of the chain (when viewed from *Home*) is for outbound traffic, and the left side for inbound traffic.

4.1.3 Extending the Chain - JoinChain

JoinChain is implemented as a combination of three behaviors: an extended **Tack**, **BackInto**, and **AlignBack**.

The extended **Tack** times the intervals between contacts with the chain. If the more than a given time passes (in our experiments, 10 seconds), it sends out a signal. This signal is used to deactivate **Tack** and activate **BackInto**.

BackInto reverses at a sharper angle than that used in forward tacking. When combined with an appropriate (empirically determined) time-out for **Tack**, this gives the robot a likelihood of contacting the front of the *EndOfChain* robot with its back bumper. If contact is not made within a certain time period (30 seconds in our experiments), it is assumed that the end of the chain has been missed and the robot continues forward at the tacking angle until something is contacted. If contact is made, the robot withdraws enough to clear contact and sends a signal which deactivates **BackInto** and activates **AlignBack**.

AlignBack delays to avoid confusion between the first contact tap of **BackInto** and its own communicative taps, then taps the *EndOfChain* robot twice (within three seconds), adjusting its angle relative to the *EndOfChain* according to contact indication through left, right, or both rear contact switches. If the *EndOfChain* responds with two taps (within three seconds), the robot is fairly well aligned and considers itself to have joined the chain. In doing so, it deactivates, and takes the role of *MiddleOfChain* (there is currently no specialization required for the *EndOfChain*). If it does not receive two answering taps within a fixed interval (10 seconds), it circles left at the tacking angle until it hits something, then deactivates **AlignBack** and activates **Tack**.

4.1.4 MiddleOfChain - Link

MiddleOfChain has been actualized as the behavior **Link**, which detects and responds to double taps to its front and rear. This corresponds to A-F of Figure 2, and is also enough to satisfy the requirements of the *EndOfChain*. Time-outs on tap attempts to the front and single contacts with the previous chain link allow recovery from most errors.

4.1.5 Excursion Search

Excursion Search has been implemented through an extended **Tack** and a behavior **CircleRight**. The extended **Tack** makes a decision every time it contacts the

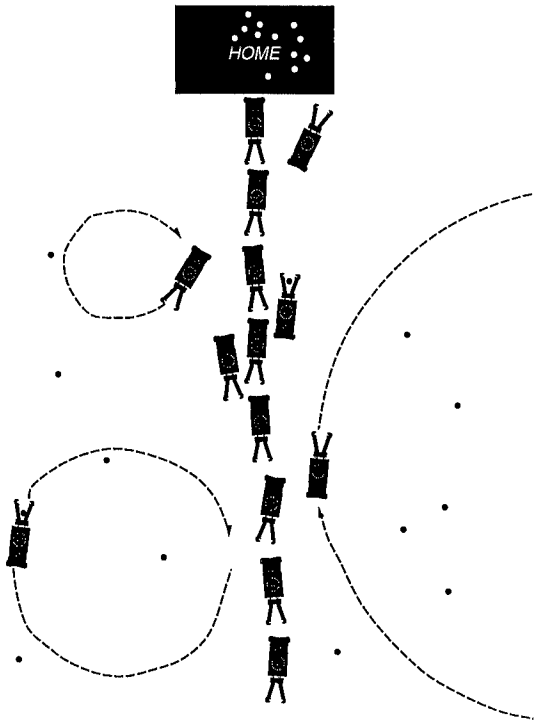


Figure 4: Excursion Search strategy: Robots search for pucks and return to the chain by making roughly circular "excursions" from the chain.

chain as to whether or not it should make a circular excursion to the right to search for pucks. No excursions are made if the robot is already holding a puck, otherwise the choice is random (1/8 chance in our experiments).

4.2 The Robot Herd

Our experiments are implemented and tested on the Nerd Herd, the Interaction Lab's group of 20 IS Robotics R1 mobile robots. Each member of the Nerd Herd is a 12–inch four–wheeled vehicle, equipped with a two–pronged forklift for picking up, carrying, and stacking pucks (Figure 5). The forklift contains two contact switches, one on each tip of the fork, six infra–red sensors: two pointing forward and used for detecting objects and aligning onto pucks, two break–beam sensors for detecting a puck within the "jaw" and "throat" of the forklift, and two down–pointing sensors for aligning the fork over a stack of pucks for stacking (Figure 6). The pucks are special–purpose light ferrous metal foam-filled disks, 1.5 inches diameter and between 1.5 and 2.0 inches in height. They are sized to fit into the unactuated fork and be held by the fork magnet. Each robot also has one piezo–electric bump sensor on each side of the chassis. Only the front contact, the stacking IRs,

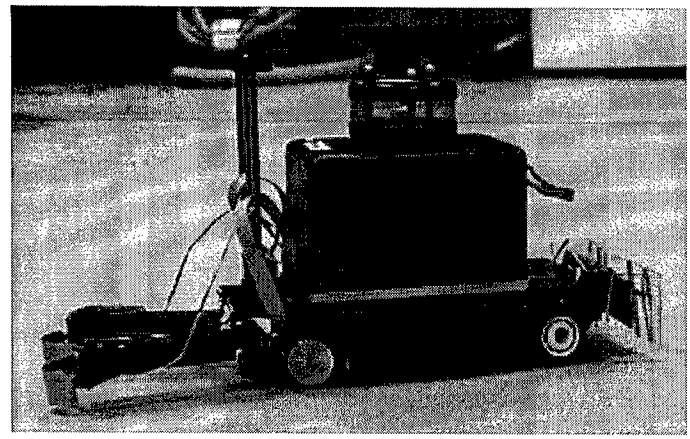


Figure 5: Each of the Nerd Herd robots is a 12"–long four–wheeled base equipped with a two–pronged forklift for picking up, carrying, and stacking pucks, and with a radio transmitter and receiver for inter–robot communication and data collection.

and rear contact sensors described in 4.2.1 are used in the described experiments.

The mechanical, communication, and sensory capabilities of the robots allow for exploration of the environment, robot detection, and finding, picking up, and carrying pucks. These basic abilities are used to construct various experiments in which the robots are run autonomously, with all of the processing and power on board. The processing is performed by a collection of four Motorola 68HC11 microprocessors. Two of the processors are dedicated to handling radio communication, one is used by the operating system, and one is used as the "brain" of the robot, for executing the down–loaded control system used in the experiments. The control systems are programmed in the Behavior Language, a parallel programming language based on the Subsumption Architecture [Brooks 1986, Brooks 1990].

4.2.1 Hardware Modifications

Originally equipped with piezo–electric bump sensors on the back of the chassis, the venerable robots are being modified to better suit the chaining task. The rear surfaces of some robots now have large bumpers that activate contact switches (see Figure 6). This is necessary due to the nature of the bump sensors, which cannot indicate continuous contact, and to the fact that the width of the original rear surface is the same as the width of the opening of the fork - which leads to constant catching and damaging of the fork-mounted contact sensors in the alignment task.

4.2.2 Hardware Limitations

As discussed in Section 1, properties of physical hardware impose restrictions not only on the control strategies that can be applied, but also on the types of tasks and experiments that can be implemented. Robot hardware is constrainted by various sensory, mechanical, and computational limitations.

Our robots' mechanical steering system, when in perfect condition, is "accurate" to within 30 rotational degrees. At certain steering angles, the drive wheel is lifted off the ground, while at others, the steering wheels jam against metal parts of the chassis. During any type of physical interaction, parts tend to change alignment.

The uncertainty and variability inherent in any work with physical robots and especially salient in the case of the R1s, although frustrating, is beneficial to experimental validity. Hardware variability between robots is necessarily reflected in their group behavior. Even when programmed with identical software, the robots behave differently due to their varied sensory and actuator properties. Small differences among individuals become amplified as many robots interact over extended time. As in nature, individual variability creates a demand for more robust and adaptive behavior. The variance in mechanics and the resulting behavior have provided stringent tests for our methodologies.

4.3 Performance of Current Implementation

The foraging system that we tested with six working robots demonstrates practicability of our robot chain concept. While some behaviors demonstrated a high failure rate, graceful recovery allowed multiple attempts, as detailed below. Ongoing software and hardware refinements are providing consistent increases in reliability, especially in regards to previously ubiquitous mechanical failures.

The ability of robots to follow the formed chains was robust, and was lost only when mechanical failures led to following robots pushing chain robots so far as to open up wide gaps in the chain. Any gap wide enough to permit the front contact sensors of a following robot to cross the chain (about 1 robot length, 12 inches, depending on the turning circle of the particular robot involved) tended to result in unrecoverable errors. The average separation in well-formed chains was observed to be about six inches, and the nature of the communication along the chain tended to maintain this distance through minor (though not major) "pushing" by following robots. The effective length of a chain can be said to be approximately 1.5 times the length of the robots that form it.

Chain following suffered only rare mechanical failures. Some of those cases resulted in the following robot changing the position of one or more of the chain robots to such degree that chain integrity was broken. Detecting and recovering from such problems is a step on the way to dynamic chain readjustment, and is currently being developed.

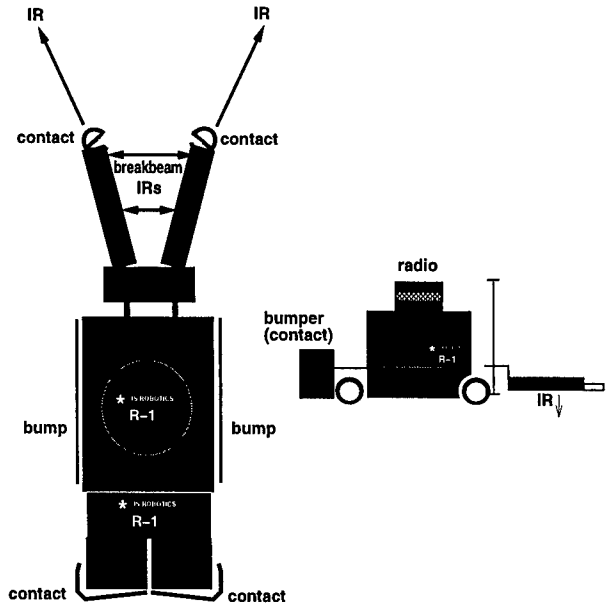


Figure 6: Each of the Nerd Herd robots is equipped with contact sensors at the ends of the fork, piezo-electric bump sensors on each side and two on the rear of the chassis, and six infra-red sensors on the fork. Two forward-pointing IRs are located at the ends of the forks, two break-beam IRs in the jaw and throat of the fork, and two down-pointing IR for stacking pucks in the middle of each of the fork arms. The result of replacing the rear piezo-electric bump sensors with bumpers and contact sensors is shown.

The only problems encountered during *Excursion Searching*, besides mechanical ones described above, occurred when a robot pushed more than two pucks at a time, which prevents the front sensors from making any contact. This problem was resolved in some trials through a time-out which backs up after a given period without contact (we actually used **Tack** and **BackInto** from *JoinChain* quite successfully). Our limited number of robots only allowed us to have one robot searching for pucks while the others formed the chain; in this case, the searcher brought an average of one puck *Home* each trip around a chain of four robots, which took about two minutes, depending on the number of circular excursions. With more functional robots we will examine the effects of interference along the chain and its influence on scalability.

The *JoinChain* process requires the most precision and was most prone to failure. Approximately fifty percent of attempts made a successful first contact (in **BackInto**),

and of these approximately fifty percent exchanged taps and resulted in joining the chain. These rates could be improved by tuning the steering systems of the robots and/or tuning the timing of individual robots, but improvement would be only temporary since alignment changes rapidly. Though a raw success rate of twenty five percent does not seem impressive, graceful recovery and persistence of attempt allowed eventual joining in most cases. This is exactly the type of trade-off we intend: that a large number of less capable, more robust, somewhat expendable agents can perform certain tasks at least as efficiently as a smaller number of more sophisticated agents. As in natural systems, such as ant pheromone trail formation, global behavior is a result of the cumulative effects of many actions. The key point we see in both natural (i.e., ant) and artificial (i.e., robotic) systems is that while individual successes benefit the system as a whole, individual failures do not accumulate. The most efficient ant paths are more frequently traveled than the longer ones, and are thus given a stronger marking that overpowers, and outsurvives, the weaker ones. Analogously, in robot chains, only those robots that successfully join the chain have a lasting effect on the behavior of others. In both systems, success results in a persistent encoding of information in the environment, while failure does not.

5 Discussion

[Donald et al 1994] demonstrated the utility of their theoretical framework of information invariants in analyzing tradeoffs and equivalences between sensor systems. Specifically, they showed the reducibility of one system that used explicit communication between two robots to one that did not (i.e., which communicated solely through task dynamics). They also raise the following questions: 1) "can robots "externalize," or record state in the world?" and 2) "can we record "programs" in the world in the same way we may externalize state?" Our research addresses these questions with a system of robots that form distributed physical representations of spatial information. Where [Donald 1995] discusses "calibrations" of sensor systems which fix certain spatial relationships (effectively encoding spatial information) in the system, we present a system that continuously calibrates itself to encode changing information into a distributed representation of spatial relationships, or, in other words, to continuously re-engineer the environment so as to influence the behavior of individual agents. Since these physical representations direct the behavior of agents within the system, they may be seen as "programs" that the system as a whole encodes into the environment for "execution" by its parts.

Practically, we can see that such externalization of state and control allows a wider range of robots - particularly, much simpler robots - to perform various classes of tasks. Through collective behavior, local (at the extreme, physical contact) sensors can suffice for tasks that require global position information. Future research will begin the process of extending the information invariants-based analysis and develop the existing design methodologies to encompass the notion of dynamic self-calibration.

More philosophically, in externalizing more and more of the cognition required to perform any task, we shift our focus farther from intra-agent processing and further towards interaction between agents. The extreme simplification of control within an agent allows us to locate interesting behavior at this level of interaction. Since these interactions are all physical and observable, our vantage point for observation of "emergent" behavior is substantially improved.

6 Conclusion

We have shown that chains of robots using only physical contact-range sensing can solve certain global position-dependent problems. This contradicts a heretofore assumed need for more complicated sensors, positioning systems, or processing. Many environments and applications (especially a number of those proposed for development of "nanorobot swarms", undersea exploration, and space exploration), due to size and/or ambient noise factors, impose exactly these types of restrictions on position–dependent tasks. Systems similar to that described here should drop the lower bound on hardware (and therefore cost) requirements for a wide range of position–dependent tasks, and extend the range of environments in which they are possible.

Some robotics research has presented or reproduced particular instances of *stigmergy* - "the production of a certain behavior in agents as a consequence of the effects produced in the local environment by previous behavior" [Beckers et al 1994] (see also, for example, [Deneubourg et al 1991], [Theraulaz et al 1991]) - but analysis has remained at the level of claims of greater robustness or ease of scalability than an often undescribed "centralized" system. Many proposed robotic applications are poised to take advantage of these properties of stigmergy, but must wait for a better understanding of what the systems *can* do, and likely the ability to make some guarantees about what the systems *will* do. The robot chaining system is one example of a deliberate and useful exploitation of stigmergic effects that we hope will serve as inspiration and object of analysis for development of methodologies for externalization.

Acknowledgements

The research reported here was done at the Interaction Laboratory at the Brandeis University Volen Center for Complex Systems and the Computer Science Depart-

ment. The work is supported by the Office of Naval Research Grant N00014-95-1-0759 and the National Science Foundation Infrastructure Grant CDA-9512448.

The authors thank Jaroslav Hook, Francisco Mello Jr., and Lester Lehon for their contributions to the functionality of the Nerd Herd, Dani Goldberg for emergency cool, Jordan Pollack for getting his hands dirty, and Pablo and Karina Funes for the *remedio*.

References

[Aron et al in press] "Functional Self-organization Illustrated by Inter-nest Traffic in Ants: The Case of the Argentinian Ant," S. Aron, J.L. Deneubourg, S. Goss, J.M. Pasteels, *Biological Motion , Lecture Notes in BioMathematics*, W. Alt and G. Hoffman, eds.

[Beckers et al 1994] "From Local Actions to Global Tasks: Stigmergy and Collective Robotics" R. Beckers, O.E. Holland and J.L. Deneubourg, *Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems*, 181-189.

[Brooks 1986] "A Robust Layered Control System for a Mobile Robot", Rodney A. Brooks, *IEEE Journal of Robotics and Automation*, RA-2, April, 14-23.

[Brooks 1990] "The Behavior Language; User's Guide", Rodney A. Brooks, *MIT A.I. Lab Memo 1227*, April.

[Deneubourg et al 1991] "The Dynamics of Collective Sorting: Robot-Like Ants and Ant-Like Robots", J.L. Deneubourg, S. Goss, N. Franks, A. Sendova-Franks, C. Detrain and L. Crétien, *Proceedings of the First International Conference on Simulation of Adaptive Behavior: From Animals to Animats*, MIT Press, 356-363.

[Deneubourg et al 1990] "Self-organizing collection and transport of objects in unpredictable environments," J. Deneubourg, S. Goss, G. Sandini, F. Ferrari, and P. Dario, *USA-Japan Symposium on Flexible Automation*, Kyoto, Japan, July.

[Donald et al 1994] "Information Invariants for Distributed Manipulation," B. Donald, J. Jennings, and D. Rus, *The First Workshop on the Algorithmic Foundations of Robotics*, R. Wilson and J.-C. Latombe, eds. A.K. Peters.

[Donald 1995] "Information Invariants in Robotics," B. R. Donald, *Artificial Intelligence* 72.

[Fontán and Matarić 1996] , "A Study of Territoriality: the Role of Critical Mass in Adaptive Task Division," Miguel Schneider Fontán and Maja J. Matarić. *Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior*, Cape Cod, September.

[Goldberg Matarić 1996] "Interference as a Guide for Designing Efficient Group Behaviors'," Dani Goldberg and Maja J Matarić. Brandeis University Computer Science Technical Report CS-96-186.

[Gomi 1995] "Non-Cartesian Robotics", Takashi Gomi, *Proceedings of the International Conference on Biorobotics: Human-Robot Symbiosis.*

[Goss et al 1989] "Self-organized Shortcuts in the Argentine Ant," S. Goss, S. Aron, J.L. Deneubourg, and J.M. Pasteels, *Naturwissenschaften* 76.

[Goss and Deneubourg 1991] "Harvesting By A Group Of Robots," S. Goss, J. L. Deneubourg, *Proceedings of the First European Conference on Artificial Life*, MIT Press.

[Hölldobler 1990] The Ants. Bert Hölldobler and Edward O. Wilson, Cambridge, Massachusetts: The Belknap Press of Harvard University Press.

[Horswill 1993] "Specialization of Perceptual Processes", Ian D. Horswill, MIT PhD Thesis, May.

[Matarić 1992b] "Integration of Representation Into Goal-Driven Behavior-Based Robots," Maja J Matarić, *IEEE Transactions on Robotics and Automation*, 8:3, June.

[Matarić 1990] "Navigating with a Rat Brain: A Neurobiologically-Inspired Model for Robot Spatial Representation", Maja J Matarić, *Proceedings of the First International Conference on Simulation of Adaptive Behavior: From Animals to Animats*, MIT Press.

[Matarić 1995] "Designing and Understanding Adaptive Group Behavior," Maja J Matarić, *Adaptive Behavior* 4:1, December.

[McFarland 1994] "Towards Robot Cooperation", David McFarland, in *Proceedings of the Third Annual Conference on Simulation of Adaptive Behavior: From Animals to Animats*, D. Cliff, P. Husbands, J. Meyer, and S. Wilson, eds., Cambridge, MA, MIT Press.

[Russell 1995] "Laying and Sensing Odor Markings as a Strategy for Assisting Mobile Robot Navigation Tasks," R. Andrew Russell. *IEEE Robotics and Automation Magazine*, September.

[Steels 1994] "A Case Study in the Behavior Oriented Design of Autonomous Agents,"Luc Steels, in *Proceedings of the Third Annual Conference on Simulation of Adaptive Behavior: From Animals to An-*

imats, D. Cliff, P. Husbands, J. Meyer, and S. Wilson, eds., Cambridge, MA, MIT Press.

[Muller and Wehner 1988] "Path Integration in Desert Ants: Cataglyphis Fortis," Martin Muller and Rudiger Wehner. *Proceedings of the National Academy of Sciences.*

[Theraulaz et al 1991] "Task differentiation in Polistes wasp colonies: a model for self-organizing groups of robots," Guy Theraulaz, Simon Goss, Jacques Gervet, and Jean-Louis Deneubourg. *Proceedings of the First International Conference of Simulation of Adaptive Behavior: From Animals to Animats.*

(Not) Evolving Collective Behaviours in Synthetic Fish

Nahum Zaera[1], Dave Cliff[2], and Janet Bruten[1]

[1]Hewlett-Packard Laboratories Bristol,
Filton Road, Stoke Gifford, Bristol BS12 6QZ, U.K.

[2]School of Cognitive and Computing Sciences,
University of Sussex, Brighton BN1 9QH, U.K.

nz@hplb.hpl.hp.com davec@cogs.susx.ac.uk jlb@hplb.hpl.hp.com

Abstract

We describe a series of experiments in which artificial evolution is used to develop neural-network sensory-motor controllers for software animats which exhibit collective movement behaviours in three dimensions. We successfully evolved controllers which displayed simple group behaviours such as dispersal and aggregation. However, attempts to evolve realistic-looking schooling behaviours never succeeded. The problem appears to be due to the difficulty of formulating a scalar evaluation metric, or "fitness function" which captures what schooling is. We argue that formulating an effective fitness function for use in evolving controllers can be at least as difficult as hand-crafting an effective controller design. Although our paper concentrates on schooling, we believe that this is likely to be a general issue, and is a serious problem which can be expected to be experienced over a variety of problem domains.

1 Introduction

Artificial evolution, in the form of genetic algorithms (GAs), is often advocated as a labor-saving approach, where the design of complex artifacts can be achieved semi-automatically. Several authors have reported work in which an evolutionary process has been used to develop designs for sensory-motor "controller" coordination mechanisms for animats (i.e. artificial autonomous agents, either mobile robots or virtual agents).

In a recent review of selected work in this area [MAT96], it was noted that in many cases the behaviours produced by the final evolved controllers were relatively simple, and that controllers which produce equivalent behaviours could feasibly be designed manually with no more effort than was required to construct the artificial evolution system. It was also noted that such results are to be expected at this early stage in the development of the field, where the concept of using artificial evolution is being demonstrated, and where the evolutionary techniques are being developed, refined, and extended.

In this paper, we discuss issues arising in applying evolutionary design techniques to a challenging class of problems, namely coordinated collective behaviours of groups of animats. The ultimate aim of our work was to evolve sensory-motor controllers which would give rise to behaviours similar to flocking in birds, schooling in fish, or herding in land animals. These animal behaviours have been extensively studied in the biology literature, and a number of researchers have reported results where animats (both real and simulated) have been constructed to exhibit comparable collective behaviours. Most of the work with animats has involved manually designed controllers, and the indications are that the design of such controllers is a difficult task. Artificial evolution would seem to offer a route by which much of this hard work can be avoided.

In the work we discuss here, we intended to evolve neural-network sensory-motor controllers for virtual "fish" which could move in three dimensions subject to relatively realistic "laws of physics" (i.e. drag, momentum, inertia, etc.). Hence we will talk about "schooling", but this can be taken to refer also to "flocking", "swarming", and "herding". Our work was motivated by "engineering" concerns (i.e. exploring GA's as an automated design method) rather than by "scientific" concerns (e.g. trying to understand what selection pressures lead to coordinated group behaviors in animals).

While we found it relatively easy to evolve controllers which produced simple collective behaviours such as aggregation or dispersal, we experienced significant difficulties in evolving schooling behaviours. The main aim of this paper is to highlight the problems we encountered, in the belief that they are inherent in the evolution-

ary approach, and are too often underplayed or not reported in the literature. We will argue that formulating an effective fitness evaluation function for use in evolving controllers can be at least as difficult as hand-crafting an effective controller design. Although our paper concentrates on schooling, we believe that this is likely to be a general issue, and can be expected to be experienced over a variety of problem domains.

Section 2 reviews past work in creating animats which exhibit schooling-type collective behaviours. Section 3 then discusses our evolutionary simulation system. Following this, Section 4 presents results from experiments where a variety of collective behaviours are evolved. Section 5 then discusses our results and the probable reasons for failure to evolve satisfactory schooling. Our conclusions are presented in Section 6.

2 Schooling in Animals and Animats

Fish schooling and other forms of animal aggregations (flocks, herds, and swarms) have always been one of the greatest spectacles that nature can offer. Biologists, zoologists, and lately animat researchers, have shown interest in them. Such natural exhibits are often cited as examples of emergent collective behaviour.

Emergence of collective behaviour is interesting because it offers the possibility of creating complex global behaviour from local interactions between relatively simple agents. It is the sum of these local interactions that makes the system complex as a whole.

It is beyond the scope of this paper to provide a full review of the biology literature: here we briefly discuss hypotheses concerning schooling in fish. We then present a short review of related work in animats. More detailed reviews of both areas are presented in [ZAE95].

2.1 Schooling in Fish

Biologists have proposed several different hypotheses for schooling behavior. Being in a school serves to reduce the risk of being eaten by a predator [PAR82,SHA62,PIT86], provides mating efficiency, makes finding food easier, and is a good environment for learning and reducing overall aggression [BILL76]. Some authors argue that schooling also saves energy by improved hydrodynamic performance through reducing drag. However, there are contradictory opinions and no conclusive results can be drawn [BILL76,PAR79].

Schooling seems to obey the rules of a distributed model (each individual applying the same set of simple behavioral rules). Each fish takes into account all fish that swim in its neighborhood, paying more attention to the closest ones and trying to match its velocity and direction with that of its neighbors [PAR82,SHA62]. Fish also try to maintain a constant distance between themselves and their close neighbors [BILL76].

2.2 Related Animat Work

A seminal work on schooling in animats is Reynolds' "Boids" behavioral animation system [REY87]. In this system, virtual agents exhibit schooling behaviours in 3-D. The behavioral control of each agent can be described by a set of simple rules, but each agent's controller involves a sophisticated "arbitration" mechanism. Achieving successful schooling in Boids requires fine-tuning a number of parameters in the arbitration mechanism.

Boids used incremental geometric flight to move the agents through the environment: this models conservation of momentum and viscous speed damping. Visual perception in each agent is modeled to the extent that it provides the behavior model with similar information to that available to as the end result of perceptual processes. However, the generation of such information by a perceptual or cognitive system presents a number of significant difficulties. In this sense, Boids is a "perfect information" system: the relevant variables are continuously available for each agent without noise or error.

Other work on simulation of collective behavior tends to take a similar approach. Accurate modeling of perception is the weak point of most of them, and various simplifications are usually adopted. Senses like hearing or smell have also been used as metaphors for simple sensing [WER92]. Some researchers have modeled additional aspects that affect behavior, such as internal states (hunger, fear, libido, energy level, fitness, etc.) [TU94A,TU94B,WER92]. Other authors have treated schools of fish like particle systems, concentrating on attraction and repulsion forces and dynamics [AOK82,NIW94].

Mataric [MAT92,94,95] demonstrated schooling in real robots, working with real sensors. Her research, in common with much other work in collective animat behaviour, relied on hand-crafting the agents' controllers. The schooling behaviour was created by hand-tuning weights which combined the contributions of less complex component behaviours such as aggregation and avoidance. Mataric notes [MAT92,P.438] "Due to the number of tunable parameters involved, *flocking* is the most complex basic interaction implemented in this work so far".

While much work in animat schooling is based on hand-crafting behaviours, there has been some prior work on evolving the controllers for virtual animats in the form of Lisp expressions [REY92], and simple feed-forward neural networks [WER92]. Rucker [RUC93] used an ecosystems model (i.e. multiple interacting animat spe-

cies) to tune genetically-encoded parameters which governed a schooling controller inspired by Reynolds' work, but with the arbitration mechanism replaced by a simple linear combination of control variables. Again, all the work on evolving schooling controllers with which we are familiar relies on "perfect information" control variables and hence issues in sensory processing for guidance of schooling are avoided entirely.

2.3 Summary

The following two points are of primary concern to our work:

First, the use of perfect information (where particular relevant variables are chosen a priori as noise-free inputs to the controller) in virtual animats raises the problem that a sensory system delivering such information is neither biologically plausible nor realistically implementable in real robots. Moreover, deciding which environmental variables are important is often an intuitive or heuristically-guided process. This pre-commitment to particular variables could be avoided by giving the virtual animats realistic simulated sensory systems and then allowing some adaptation process to determine which factors in the sensory input are relevant for guidance of the desired behaviours.

Second, most of the animat controllers developed to date are hand-coded. The complexity of this manual design task rises as the simulations become more realistic, and the difficulty is presumably most acute in real robot systems such as the one developed by Mataric. Such design processes tend to be difficult, heuristically guided, and time consuming. In principle, it should be possible to use artificial evolution to (semi-) automate this design process. However, the few studies using evolution to design controllers for schooling have worked on simulations with so many simplifying assumptions that their relevance is perhaps questionable.

In principle, it should be possible to use artificial evolution to develop controllers which give rise to schooling behaviours in virtual animats with sensory systems more realistic than those used in prior research. This was the aim of our experiments, described in further detail below.

3 Evolving Animats for Collective Behaviours

The animats in our experiments were loosely modeled on fish, and the aim was to evolve controllers which gave rise to collective behaviours analogous to fish schooling. The primary interest was the notion that coordinated and coherent "global" group behaviours could arise from the interaction of a number of agents, each of which has access to only "local" information (i.e. that gathered from range-limited sensors). There was no intention to create a faithful simulation of reality such as that developed by Terzopoulos et al. [TU94A,TU94B] where the fishes' mechanics and hydrodynamics are modeled.

Fish are an attractive source of inspiration for two reasons. First, fish move in three spatial dimensions. Avoiding collisions in two-dimensional schooling (e.g. "herding" in terrestrial agents) is more difficult than in the three-dimensional (aquatic or aerial) case where alterations in depth/altitude can be used as an alternative to taking evasive action within the horizontal plane. Second, fish combine visual perception and perception from the "lateral line" pressure sensors. The sensors on the lateral line, which runs longitudinally on each side of the fish, are responsive to local variations in water pressure. Typically, such pressure variations correspond to the fish swimming close to an obstacle (which may be inanimate, such as a rock, or animate, such as another fish), and the indications are that fish use lateral lines to sense both proximity and relative velocity.

The attenuation of light and pressure in water differ, such that the visual system provides distal sensory information while the lateral lines provide proximal information. This mix of distal and proximal sensors is analogous to the use of e.g. visual and tactile sensors on mobile robots.

Our animats had sensory systems which were minimalist approximations to vision and pressure sensors. The physical arrangement of these sensors was fixed during every experiment (i.e., not under evolutionary control) and lead to a fixed-connectivity architecture for the artificial neural network controller. Sensory-motor coordination was governed by these artificial neural network controllers, the weights and thresholds of which were under evolutionary control.

We used neural networks as the basis for the evolving controllers because of the widely accepted argument that their properties of graceful degradation with respect to alterations in weights and thresholds results in smoother evolutionary fitness landscapes. A number of authors have reported successful application of artificial evolution to developing a wide variety of styles of neural network [KUS94].

Further details of the synthetic fish are given in Section 3.1, with details of the evolutionary approach given in Section 3.2. Results from experiments in evolving simple collective behaviours are described in Section 3.3, and results from evolving schooling are discussed in Section 3.4.

3.1 The "Synthetic Fish" Animats

The "fish" animats exist in a 3-D space with bounds on the "depth" (Z) axis but no limits on its horizontal (XY) extent. These bounds are an idealization of fish swimming in the open sea, where they can roam freely but have to stay between the sea-surface and the sea-bed.

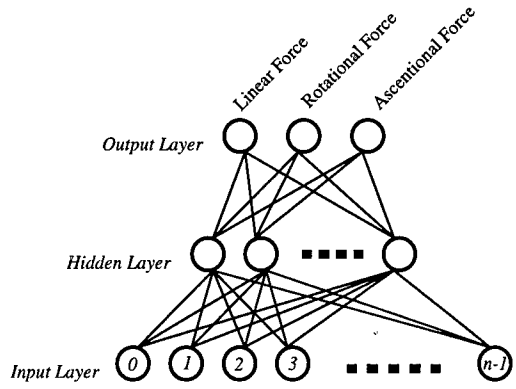


FIGURE 1. Neural network architecture.

The behaviour of each animat is determined by a three-layer feed-forward neural network. Units (i.e. "artificial neurons") in one layer are fully connected to all units in the next layer, and there are no connections within a layer. The instantaneous output of a unit is a function of the sum of its current inputs. For each unit, the connection weights and the nature of the transfer function were genetically determined: see [ZAE95] for further details. Units in the input layer were either "visual" or "lateral-line" sensors: their inputs values were determined by models of these sensory processes discussed further below.

The neural network for each animat had three output units, which were interpreted as forces which produced: linear acceleration in the Z axis; linear acceleration in the XY plane along the animat's longitudinal axis; and angular acceleration about the animat's Z ("yaw") axis (i.e. altering its orientation in the XY-plane). Simple Newtonian point-kinematics were used to determine the translational and rotational velocities of the animat as the outputs of the network varied: all animats had the same notional masses, moments of inertia, and drag coefficients. Drag coefficients give asymptotic upper limits on speeds when acceleration forces are applied indefinitely.

For the purposes of simulating sensory input, each animat was considered to have a spherical "body".

Lateral Lines

Each animat has six "pressure-sensitive" sensors located at the top, bottom, front, back, left, and right of its spherical body. See Figure 2.

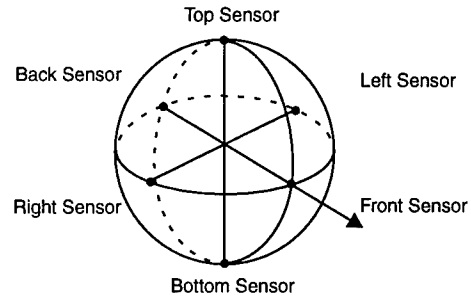


FIGURE 2. Location of "pressure sensors".

The vector from the centre of the animat's body through the location of the sensor forms a normal-vector for the "threshold-plane" of the sensor. The sensor is sensitive to all other objects (i.e. animats, and the "sea-bed" and "sea-floor" planes) on the distal side of the threshold plane (i.e. the opposite side from the animat's body). The response from each sensor is a sum of the proximity contributions from these objects. Proximity contributions from each object are attenuated with distance and with increases in the angle between the threshold-plane normal-vector and the vector from the animat to the nearest point on the object: see [ZAE95] for further details.

Vision

Most biological studies suggest that in fish, the visual system is used to maintain distance and angle to the closest neighbors. Apart from this role, which overlaps with that of the lateral lines, the visual system is used as a long distance perceptual system. In our experiments it could have a very important role at the beginning of the simulation, where depending on the initial conditions animats may be widely scattered, and a long-distance sensor is needed to bring them together. In real fish, the visual system is thought to be the primary sensory mechanism underlying the formation of fish schools from several individuals.

Our animats can "see" the world through a primitive 360-degree patched retina. The patches are directly mapped on to parts of the spherical body. Each patch has a value that depends on how many fish are in the pyramid of vision corresponding to that patch. The pyramid of vision is defined as the part of the space delimited by the segments that go from the centre of the animat through each of the vertices of the patch: see Figure 3.

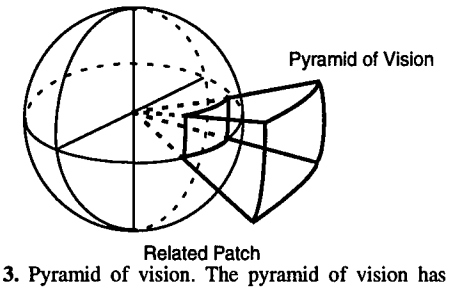


FIGURE 3. Pyramid of vision. The pyramid of vision has infinite volume as it defined by unbounded planes.

The response of a visual sensor unit is proportional to the number of animats within the sensor's pyramid of vision, with the contribution from each animat modulated by its distance from the sensor.

The number of patches was not under genetic control. In all our experiments there were nine patches in a 3x3 grid on the frontal hemisphere of the animat. We tried experiments where the rear hemisphere was a single patch, and also where it was a 2x2 grid of patches.

Having more patches (i.e. higher "resolution") at the front of the animat is an attempt to give it more biological plausibility. This approach models an approximation to foveal vision. Whether this potential advantage is exploited by the synthetic fish or not depends entirely on the evolutionary process.

3.2 The Evolutionary Process

In our experiments, the genotypes that the genetic algorithm manipulates encode the parameters for the transfer function of every unit of the neural net controller as well as the weights between connected units. The organization of the genotype is very simple. Every gene is coded as a double precision floating point number in the range [0.0,1.0). The sequence in which the units are placed in the genotype is also straightforward. Starting from the first unit of the input layer, following with the units from the hidden layer and finally the output units: see Figure 4.

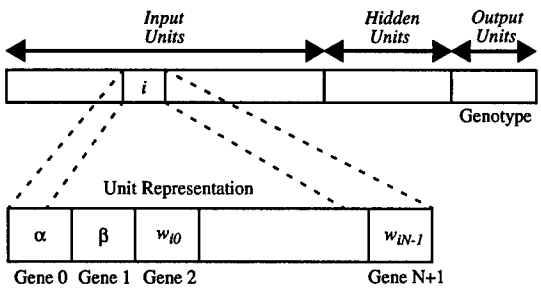


FIGURE 4. Genotype structure: α and β are parameters for the unit's transfer function.

The genetic operators used were one-point crossover and mutation. Both operators act at the gene level. The mutation operator works by adding small random (zero-mean) perturbations to one of the real values on the genotype.

A typical experiment, with nine visual patches at the front and one big patch at the back, gave a neural network configuration of 16 input units, 5 hidden units and 3 output units. Using our genetic encoding this configuration amounts to a genotype length of 143 real numbers. To set the mutation rate and crossover rate of the genetic algorithm we used information from convergence measures (average and standard deviations of the mean Euclidean distance to the elite and mean genotypes). We found it appropriate to set the mutation rate at approximately one gene mutated per genotype in the next population. Selection was rank-based, with a near-quadratic selection pressure.

As we were interested in collective behaviours from groups of homogeneous animats, we evaluated an individual genotype by monitoring the behaviour of groups of *clones* of that genotype. An individual genotype was used to set the parameters for the neural networks of a number of animats. Simulation of these animats moving in the 3D environment was then monitored to determine the fitness of the genotype.

The genetic algorithm was usually run for 100 generations with a population of 100 genotypes. We did most of our simulations with 4 or 5 animats and using only a single set of initial conditions (only one evaluation per genotype). Each simulation was run for 2000 steps.

We also tried some experiments using different settings: more animats, more sets of initial conditions, more simulated time, more generations, etc. None of these variations significantly affected the qualitative features of the results presented below.

3.3 Dispersal and Aggregation: Success

In order to confirm that our evolutionary system could operate successfully (i.e. that we had the various parameters such as mutation rate, crossover rate, etc. set correctly), we first ran experiments where collective behaviours less challenging than schooling were evolved. We evolved controllers for "dispersal" and "aggregation" behaviours. Both of these require that the animats be sensitive to the actions of others in the group. Indeed, the only significant bodies in the environment with which an individual can interact are other animats.

Dispersal

The evaluation function for dispersal was based on maximizing a measure of the distance to the nearest neighbor of every fish. On each timestep, the distance to

each fish's nearest neighbor was measured: the final fitness value was based on a temporal integral of these instantaneous measures over the duration of the trial. We experimented with integrals of group-average nearest-neighbor distances, but found best results when the integral summed the smallest closest-neighbor distance (i.e. the worst pair of individuals) in the group at each step. This behaviour was the easiest to evolve. Results from a typical dispersal experiment are shown in Figure 5.

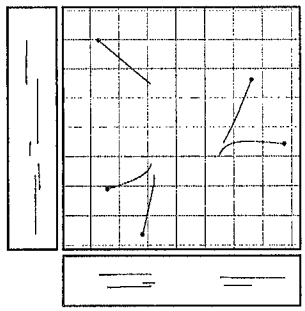


FIGURE 5. This figure shows the first 100 time steps of the dispersal experiment. Central view is an XY projection (top view), while left and bottom are YZ and XZ lateral projections respectively. When the trial starts, the animats orient away from each other and then scatter in straight-line paths until the end of the trial.

Aggregation

The aggregation evaluation function was similar to the dispersal evaluation function, but was based on *minimizing* nearest-neighbor distances. The optimal solution would be to make all the animats collide in a single point in space and not move any more. In our experiments this would have been an unlikely result, as it would have required a high degree of coordination and synchronization because if any of the animats collided with each other prematurely, their maneuverability would have been seriously impaired: collisions were inelastic. Instead, the animats converged toward (and then stayed moving in) a fairly stable bounded region of the space. This result can be considered a success because it shows a certain degree of evolutionary adaptation to the given task, in that there is an appropriate perception-reaction function coded in the neural network (we tested that by disconnecting the sensory inputs from the neural network and observing the resulting behaviour, which was similar to a random scattering behaviour). Figure 6 shows results from a typical aggregation experiment.

As we've mentioned before, for all these experiments, the space in which fish could swim was unbounded in the X-Y plane. We performed similar experiments with bounded environments (upright cylinders of different sizes). For these we added to the lateral lines sensory system the capability of detecting the proximity of the bounds of the arena as if they were any other object or fish in the simulation.

We discovered that the genetic algorithm found a way of maximizing some of our evaluation functions by exploiting the fact that the environment was bounded. An example of this is a primitive aggregation behaviour that we observed, where the fish aggregated by all heading for the centre of the environment. Even though it looked like emergent collective behaviour, we discovered that it was due to the individual interactions of each fish with the environment (bounds of the arena) rather than with the other animats.

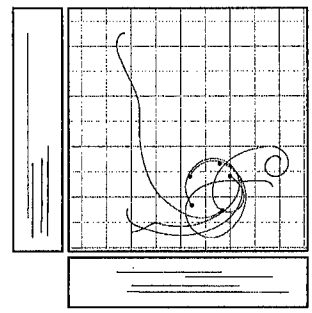


FIGURE 6. This figure shows the traces of the animats over the first 400 time-steps. From initial random positions, the animats converge to occupy a bounded region of space. Once aggregated, they all move on approximately concentric circular paths, as if they were "chasing" one another: the start of this can be seen here.

3.4 Schooling: Failure?

To evolve schooling behavior we had to provide a quantitative scalar evaluation function that measured the degree to which a group of individuals was schooling. We were unable to find a useful quantitative scalar schooling metric in the scientific literature: measures based simply on nearest-neighbor distance (e.g. [HAM71]) are really measures of aggregation rather than schooling. Mataric [MAT94,P.86] gives a formal definition of 2D schooling/flocking expressed in mathematical notation, but again this is not a scalar metric: it can only be used to make a binary decision as to whether schooling has occurred or not.

Most authors suggest that schooling is the property of moving in the same direction, at roughly the same speed and at a preferred distance from your nearby neighbors. Following those principles we formulated a variety of evaluation functions which we believed would reward appropriate behaviours.

Most of our evaluation functions involved maximizing a temporal integral of sums of instantaneous values from sub-functions that rewarded the component behaviours which other authors have proposed as combining to form schooling. The sub-functions gave rewards for "maintaining the preferred distance to neighbors" or "travelling at the same speed as neighbors", and so on. The individual rewarding functions were based on Gaus-

sian functions that returned peak values when the given schooling variable (e.g. relative distance or speed) was closest to the pre-set desired value.

The wide variety of behaviours that we obtained ranged from some fairly random-looking behaviours, through curious fixed behaviours such as every fish swimming in small circles (due to constant outputs in the neural network), to the one shown in Figure 7, where a form of schooling has actually evolved. This example was rewarded highly by our evaluation function, but it lacks realism: the two schools swim in circles. Unfortunately, this kind of primitive schooling was far from our 'folk' (i.e. subjective) notion of "realistic" schooling.

The biological literature indicates that there are factors (migratory urges, gradients in temperature or light, etc.) that make animals school in a certain direction rather than in any other. In our very impoverished environment, where the only other thing that you can detect is other animats, there is no reason to move in any particular direction. Providing some biologically plausible environment properties that the animats could detect (via appropriate sensors) might make it easier to evolve realistic schooling. More simply, a "goal" location or vector could be added in a manner similar to Mataric's 'home-vector' or Reynolds' 'leader-boid'. But both fish and birds do also exhibit non-directed schooling (i.e., where the group exhibits schooling but there is no single long-term direction of movement). Moreover, there are other possible reasons for our lack of success, discussed in the next section.

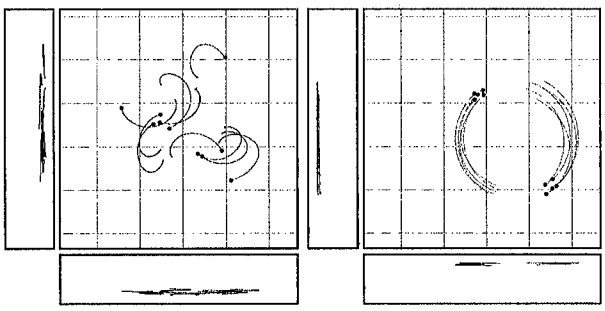


FIGURE 7. The left-hand figure shows the starting positions and the first 100 time steps of a schooling simulation. The genotype producing this behaviour was the elite from the final generation. Dots represent the positions of the animats at time step 100. The right-hand figure shows the same experiment after 1800 simulated time steps. Only the traces for the last 100 time steps are shown. Here we can see that the animats have formed two schools and swim side by side in circles.

4 Discussion: What went wrong?

Our results seem to indicate that the evolutionary approach we used is only capable of generating very simple collective behaviours. We therefore had to consider the possibility that some part(s) of our approach were unsuited to evolving more complex behaviours. The main possibilities that concerned us were: that the neural network architectures, and the nature of the sensory system, were inappropriate for generating schooling behavior; and that the evaluation function was wrong.

4.1 The Controller Architecture

We had a strong suspicion that feed-forward neural net architectures like ours are not sufficient to process the kind of information that the sensory system produces. The sensory information is effectively a "snapshot" of the surrounding environment at a given time, and the feedforward networks, having no way of maintaining internal state from one "snapshot" to the next, cannot derive differential information. Differential information, such as the rates of approach and headings of other objects, might be essential for an animat to show schooling behaviours. To calculate this kind of information, a recurrent neural network rather than a feed-forward one might be needed. Alternatively, the current sensors could be augmented with new sensors which provide differential input, such as "image motion" in the visual system.

For this reason, we replaced the sensory systems described above with a "perfect information" system similar to that used in the flocking software animats described in Section 2.2. Now the inputs to the networks were values such as the nearest neighbors rate of approach and distance. None of these experiments gave results significantly better than the earlier experiments. From this we concluded that the lack of differential information was not the (sole) reason for failure.

4.2 The Evaluation Function

In order to check our evaluation function, we implemented hand-crafted animat controllers based on rules argued to be capable of generating schooling behaviour. We developed a prototype partly inspired by Rucker's ideas [RUC93] but with some conceptual modifications needed to adapt it to our more realistic kinematics model. The appeal of this prototype is that it is a simple system, which appears to produce behaviours comparable to Reynolds' more complex Boids controller. It uses a mechanism where the response of an individual animat is based on a weighted combination of a small number of 3D vectors. Alterations to the animat's tangent vector (i.e. its direction of movement, or heading) are based on two other vectors: the nearest neighbors tangent vector and the vector from the animat to a centroid point. Alterations in speed are determined solely by the distance to

the nearest neighbor.

We manually fine-tuned all the parameters of our version of the Rucker controller, until the behavior of the animats gave what we (subjectively) judged to be realistic schooling. We then analyzed the performance of our evaluation function by examining the contributions of the sub-functions to the final fitness measure, and by checking that this good schooling behavior scored close to the theoretical maximum score. The scores of some sub-optimal controllers were also monitored, to estimate the smoothness of the evaluation function.

While this allows us to check that a given evaluation function rewards appropriate behavior and gives intermediate scores to intermediate behaviours, unfortunately it does not allow us to ensure that the evaluation function won't reward something that is not schooling: the fitness landscape generated by the function will have a peak where schooling behaviors are generated, but it may have other peaks corresponding to non-schooling behaviors. Quite possibly, these other peaks will be higher or easier to climb.

One significant difficulty is that, in contrast to evolving controllers for solitary agents, we need to develop evaluation functions which give a global measure of schooling from the observed behavior of a group of individuals. Although schooling is defined as a collective phenomenon, monitoring schooling is fundamentally based on monitoring the behavior of individuals relative to the rest of the group. Thus, there are at least four important decisions to be made in developing an evaluation function. The first concerns which individuals in the group are monitored; the second concerns which variables are monitored for those individuals; the third concerns how those variables are combined to give some instantaneous measure of group behavior; and the fourth concerns how the instantaneous measure is integrated over the duration of the trial to arrive at the final global measure of group performance. At each of these decision stages there are potentially many possible alternative approaches, and often there is nothing more than heuristic guidance as to which one to choose. So the space of plausible evaluation functions is vast.

Michalewicz [Mic94] distinguishes between multimodal optimization problems (more than one optimum) and multiobjective problems (optimize more than one thing). Our problem basically belongs to both categories. To form a school from a scattered group of animats, the group must first aggregate and then orient to the same direction. Thus the evolution of schooling is multiobjective in that the initial generations are likely to be selected primarily for their ability to aggregate. There is a danger that this behaviour dominates over the others. We have observed this with some of our evaluation functions.

We performed some experiments where we used dynamically incremental evaluation functions, where additional sub-functions were activated at different stages in an individual trial only when certain conditions were reached. For example, while the group were in the process of aggregating, measures of aggregation were weighted more heavily in the overall evaluation function than were those for schooling; once aggregated, schooling was given more significance than aggregation. Again, finding an appropriate balance between the different contributions in the evaluation function proved to be the limiting factor.

5 From a Significant Example to a General Principle?

If a standard GA is to be used to evolve schooling behaviours in animats, the evaluation function must be some quantitative scalar measure of schooling. To the best of our knowledge, nowhere in the literature on collective behaviours (in either animals or animats) is such a quantitative measure employed: the observed behaviours are described as schooling, flocking, or herding on the basis of appeal to intuitive and subjective notions of these behaviours.

It could be argued that the controllers for Mataric's robots or Reynolds' Boids constitute implicit definitions: i.e., if a group of agents do what the robots or boids do, then the group is schooling. Mataric's definition of flocking allows for a given spatio-temporal pattern of activity to be classified as "schooling" or "not-schooling", but this is not sufficient for the needs of an evolutionary approach: to apply a standard GA, a quantitative function is required which gives a graded response that in some way reflects the *degree* to which the observed behavior can be classified as schooling.

Formulating a quantitative function is not difficult: so long as the function yields a reasonably smooth fitness landscape, the GA is likely to find controller architectures which are significant improvements on the initial random designs. But formulating an *effective* function can be very hard: even when we first checked our evaluation functions against the Rucker-based controllers, the indications from subsequent evolutionary experiments were that many functions which gave appropriate rewards to schooling also gave rewards to manifestly degenerate behaviours: the space of possible behaviours satisfying the functions was not sufficiently constrained. Introducing extra constraints often resulted in oversevere fitness landscapes, where improvements on initial random designs rarely occurred.

Of course, the fitness landscape in a particular artifi-

cial evolution experiment is not solely determined by the evaluation function: it is the result of an interaction between the evaluation function and other factors such as the genetic encoding (which determines the space of possible controller architectures) and the agents' sensory-motor interactions with the environment(s) (which determine the space of possible behaviours, the ultimate "phenotypes" in the system). Nevertheless, the fact that we could successfully evolve controllers which produced simpler collective behaviours such as aggregation and dispersal indicates that it was reasonable to expect schooling to also be evolvable.

It could plausibly be argued that real schooling behaviours in real animals arise because of the complex interaction of a number of factors, and that our approach failed because the simulations lacked sufficient complexity. We have much sympathy for this argument, and other authors have already demonstrated that more complex evolutionary simulations can show interesting results. Two promising developments in this direction are Reynolds' [REY92] use of a hard-wired "predator" animat to select for controllers giving coordinated group motion in "prey" animats, and Rucker's [RUC93] use of interactions within an "ecosystem" to genetically tune parameters for his Boids-like animat controllers. However, in the absence of a schooling metric, it is difficult to judge to what extent the work of either of these authors can be regarded as the evolution of schooling: neither author attempted to quantify the degree to which their evolved animats were exhibiting schooling behaviours.

So perhaps our approach failed to capture the complexity necessary for the successful evolution of schooling. If, in order to evolve controllers for a behaviour as intuitive and simply stated as schooling, it is necessary to construct a complex ecosystem, then so be it. But, even with a more complex system, the absence of a quantitative measure of schooling implies that deciding whether schooling is occurring or not becomes a task that requires *manual* monitoring of the evolutionary process. So although no human labor is involved in designing the controllers, human labor will be needed (perhaps for days or weeks) to watch out for the possibility that the evolving animats have started to school.

Furthermore, if our lack of success was a consequence of lack of complexity, then this has significant negative implications for arguments advocating artificial evolution as a labor-saving alternative to manual design. The hand-crafted controllers in Reynolds' Boids and Mataric's robots are the result of skill and creativity on the part of their designers. In particular, Reynolds' Boids can operate in low-complexity environments without noise, perturbations, or obstacles: it seems reasonable to attempt to evolve a similar system. In principle, or so the story goes, the need for skill and creativity can be reduced by using artificial evolution: it is necessary merely to formulate an evaluation function and apply a GA. The problem our work highlights is that developing an appropriate evaluation function can be a difficult, time-consuming, heuristically-guided process requiring skill and creativity too. Possibly, just as much (or more) skill and creativity is needed with the design-by-GA approach than with the design-by-hand approach.

This is compounded by the fact that, to test an evaluation function, it is generally necessary to perform more than one evolutionary experiment with that function. As soon as the function can be demonstrated to produce satisfactory results, it can be declared a success. In the absence of a success, however, it is generally necessary to perform more than one experiment: in principle one should continue generating failures from independent experiments until statistically significant conclusions can be drawn, and each experiment should be run for long enough to give a realistic chance for successful evolution to occur. Of course, this brute force approach can be avoided if analysis of the outcome of the failed experiments reveals *why* the evaluation function is inappropriate, but such analysis can be a difficult and time-consuming task. The point here is that this "meta-evaluation" process (i.e. evaluating the effectiveness of the evaluation function) can take a lot of time and effort: again, perhaps more so than if a manual-design approach had been taken.

6 Conclusions

Our overall conclusion is that, for schooling behaviours at least, the time and effort taken to develop an evolutionary system is probably more than the time taken to develop a similar controller using manual design techniques. This paper has concentrated on what are essentially negative results. But negative results are still results. Our primary message is one of caution: our reasons for failing to successfully evolve controllers analogous to those hand-crafted by other authors are, we believe, rooted in the difficulties of formulating an effective evaluation function. This aspect of the application of artificial evolution to the development of sensory-motor controllers for autonomous agents is currently much more of an art than a science; guided as it is by heuristics and time-consuming trial-and-error techniques. Given the relative recency of the development and application of these techniques, it is perhaps no surprise to find such an important aspect of the field at a pre-theoretical stage. Interest in artificial evolution as a replacement to manual engineering-design techniques is likely to increase significantly when the evolved systems require less human

effort to create than would have been the case if they had been designed by hand. For this to happen, we anticipate, it will be necessary for the methodology to be advanced beyond the pre-theoretical stage, to the point where there are "predictive engineering" practices that can be deployed to (help) formulate the evaluation function in much the same way that an architectural engineer can (hopefully) predict whether a certain arrangement of walls will support a certain style of roof. How such practices might be developed is, at present, an open question.

Acknowledgments

Thanks to Maja Mataric and two anonymous reviewers for their thoughtful comments on earlier versions of this paper.

7 References

[AOK82] Aoki I. (1982) "A simulation study on the schooling mechanism in fish". *Bulletin of the Japanese Society of Scientific Fisheries* 48. Pages 1081-1088.

[BIL76] Bill R. G. and Hermkind W. F. (1976) "Drag reduction by formation in spiny lobsters". *Science* 193. Pages 1146-1148.

[HAM71] Hamilton, W. D. (1971) "Geometry for the selfish herd" *Journal of Theoretical Biology* Pages 295-311.

[KUS94] Kuscu I. and Thornton C. (1994) "Design of artificial neural networks using genetic algorithms: Review and prospect" in *Proceedings of Third Turkish Symposium on Artificial Intelligence and Neural Networks*, Bozsahin, C. ate al. editors. Pages 411-420.

[MAT92] Mataric M.J. (1992) "Designing Emergent Behaviors: From Local Interactions to Collective Intelligence". in *From Animals to Animats 2: Proceedings of the Second International Conference on Simulation of Adaptive Behavior (SAB92)*, J.-A. Meyer, H. L. Roitblat and S. W. Wilson editors, MIT Press, Cambridge, Massachusetts. Pages 432-441.

[MAT94] Mataric M.J. (1994) "Interaction and Intelligent Behavior" MIT EECS Ph.D. Thesis, MIT AI Lab Technical Report AITR-1495.

[MAT95] Mataric M.J. (1995) "Designing and Understanding Adaptive Group Behaviors" *Adaptive Behavior* 4(1):50-81.

[MAT96] Mataric M.J. and Cliff D. (in press 1996) "Challenges In Evolving Controllers for Physical Robots", in "Evolutional Robotics", special issue of *Robotics and Autonomous Systems*, also Brandeis University Computer Science Technical Report CS-95-184.

[MIC94] Michalewicz Z. (1994) *Genetic Algorithms + Data Structures = Evolution Programs*. Second Edition. Springer Verlag, New York 1994.

[NIW94] Niwa H.S. (1994) "Self-organizing Dynamic Model of Fish Schooling". *Journal of Theoretical Biology* 171. Pages 123-136.

[PAR79] Partridge B.L. and Pitcher T.J. (1979) "Evidence against a hydrodynamic function of fish schools". *Nature* 279. Pages 418-419.

[PAR82] Partridge B.L. (1982) "The structure and function of fish schools". *Scientific American* 246. Pages 90-99.

[PIT86] Pitcher T.J. (1986) "The functions of shoaling behaviour". in Pitcher TJ (eds.) *The Behaviour of Teleost Fishes*. Croom Helm, London. Pages 294-337.

[REY87] Reynolds C.W. (1987) "Flocks, Herds, and Schools: A Distributed Behavioral Model". *ACM Computer Graphics*, 21(4) (SIGGRAPH '87 Conference Proceedings). Pages 25-34.

[REY92] Reynolds C.W. (1992) "An Evolved, Vision-Based Behavioral Model of Coordinated Group Motion". in *From Animals to Animats 2: Proceedings of the Second International Conference on Simulation of Adaptive Behavior (SAB92)*, J.-A. Meyer, H. L. Roitblat and S. W. Wilson editors, MIT Press, Cambridge, Massachusetts. Pages 384-392.

[RUC93] Rucker R. (1993) *Artificial Life Lab.* The Waite Group Press. Corte Madera. CA.

[SHA62] Shaw E. (1962) "The schooling of fishes". *Scientific American* 206: Pages 128-138.

[TU94A] Tu X. and Terzopoulos D. (1994) "Artificial Fishes: Physics, Locomotion, Perception, Behavior". *Proceedings of ACM SIGGRAPH'94*, Orlando, FL, July, 1994, in *ACM Computer Graphics Proceedings*. Pages 43-50.

[TU94B] Terzopoulos D., Tu X. and Grzeszczuk R. (1994) "Artificial Fishes with Autonomous Locomotion, Perception, Behavior, and Learning in a Simulated Physical World". in *Artificial Life IV: Proc. of the Fourth International Workshop on the Synthesis and Simulation of Living Systems*, R. A. Brooks and P. Maes, editors, Cambridge, MA. Pages 17-27.

[WER92] Werner G.M. and Dyer M.G. (1992) "Evolution of Herding Behavior in Artificial Animals". in *From Animals to Animats 2: Proceedings of the Second International Conference on Simulation of Adaptive Behavior (SAB92)*, J.-A. Meyer, H. L. Roitblat and S. W. Wilson editors, MIT Press, Cambridge, Massachusetts. Pages 384-392.

[ZAE95] Zaera N. (1995) *Simulated Schooling: Collective Behaviours in Synthetic Fish.* Unpublished MSc Thesis. School of Cognitive and Computing Sciences, University of Sussex, Brighton.

AUTHOR INDEX

Albus, James S. 23
Araujo, E. G. 333
Arbib, Michael A. 353
Atema, Jelle 104
Aubé, Michel 264
Balkenius, Christian 305
Barnes, David P 599
Bartling, Christian 84
Beer, Randall D. 393, 421
Belanger, Jim H. 155
Bennett, Forrest H, III 430
Blanchet, Pascal 343
Blumberg, Bruce M. 295
Blythe, Philip W. 13
Bonabeau, Eric 537
Bourcier, P. de 590
Bruckhoff, Carsten 113
Bruten, Janet 635
Bülthoff, Heinrich H. 449
Chang, Carolina 373
Cliff, Dave 506, 608, 635
Cogne, François 537
Consi, Thomas 104
Corbacho, Fernando J. 353
Coronado, Juan López 373
Cribbs, H. B., III 497
Cruse, Holk 84
Dean, Jeffrey 84
Dellaert, Frank 393
Dickinson, Jeffrey 193
Digney, Bruce L. 363
Dinh, Chon Tam Le 245
Donnart, Jean-Yves 204
Duchon, Andrew P. 224
Dyer, Fred 193
Eggenberger, Peter 440
Ferrell, Cynthia 94
Floreano, Dario 402
Fontán, Miguel Schneider 553
Fuentes, Olac 382
Fukumoto, Kenji 618
Gaudiano, Paolo 373
Getz, Wayne M. 45
Ghanea-Hercock, Robert 599
Giese, Martin 113
Grasso, Frank 104
Gremyatchikh, Vera A. 173
Grupen, R. A. 333
Hallam, John 75
Hayes, Gillian 568
Hemelrijk, Charlotte K. 545
Holland, Owen 55
Horswill, Ian 163
Huber, Susanne A. 449
Humphrys, Mark 135
Ikeda, Osamu 618
Juillé, Hugues 526
Kindermann, Thomas 84
Lachiver, Gérard 245
Lambrinos, Dimitrios 65
Maes, Pattie 295
Malaka, Rainer 45
Mallot, Hanspeter A. 449
Marjanovic, Matthew 35
Mataric, Maja J 553, 625
Mayley, Giles 458
McCallum, Andrew Kachites 315
McFarland, David 255
Melhuish, Chris 55
Meyer, Jean-Arcady 204
Michaud, François 245
Miikkulainen, Risto 468
Miller, Geoffrey F. 13, 506
Mondada, Francesco 402
Moriarty, David E. 468
Moukas, Alexandros 568
Mountain, David 104
Nepomnyashchikh, V. A. 173
Neven, Hartmut 113
Noble, Jason 608
Ono, Norihiko 618
Peremans, Herbert 214
Pfeifer, Rolf 3
Pipe, A. G. 233
Pollack, Jordan B. 526, 580
Rao, Rajesh P. N. 382
Saksida, Lisa M. 285
Salomon, Ralf 411
Saunders, Gregory M. 580
Scassellati, Brian 35
Scheier, Christian 65
Schmidhuber, Jürgen 516
Schmitz, Josef 84
Schmitz, Stefan 45
Schumm, Michael 84
Senteni, Alain 264
Smith, Robert E. 497
Spector, Lee 476
Spier, Emmet 255
Steels, Luc 562
Steinhage, Axel 113
Stoffel, Kilian 476
Todd, Peter M. 13, 295
Touretzky, David S. 285
Ulbricht, Claudia 180
Veelaert, Peter 214
Ventrella, Jeffrey 484
Wagner, Hendrik 84
Webb, Barbara 75
Werger, Barry Brian 625
Williamson, Matthew M. 35, 124
Willis, Mark A. 155
Wilson, Stewart W. 325
Winfield, A. 233
Wright, Ian 272
Zaera, Nahum 635
Zalama, Eduardo 373
Zhao, Jieyu 516
Ziemke, Tom 145