ARL-TR-8407 ● JULY 2018

US Army Research Laboratory

A Cognitive Robotics System: The Symbolic and Subsymbolic Robotics Intelligence Control System (SS-RICS)

by Troy Dale Kelley and Eric Avery

NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer’s or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

ARL-TR-8407 ● JULY 2018

US Army Research Laboratory

A Cognitive Robotics System: The Symbolic and Subsymbolic Robotics Intelligence Control System (SS-RICS)

by Troy Dale Kelley and Eric Avery
Human Research and Engineering Directorate

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED (From - To)
July 2018	Technical Report	October 2003–2006

4. TITLE AND SUBTITLE
A Cognitive Robotics System: The Symbolic and Subsymbolic Robotics Intelligence Control System (SS-RICS)

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Troy Dale Kelley and Eric Avery

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
US Army Research Laboratory
ATTN: RDRL-HRF-D
Aberdeen Proving Ground, MD 21005

8. PERFORMING ORGANIZATION REPORT NUMBER
ARL-TR-8407

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT
This report details the progress on the development of the Symbolic and Subsymbolic Robotics Intelligence Control System (SS-RICS). The system is a goal-oriented production system, based loosely on the cognitive architecture the Adaptive Control of Thought - Rational (ACT-R), with some additions and changes. We have found that to simulate complex cognition on a robot, many aspects of cognition (long-term memory [LTM], perception) needed to be in place before any generalized intelligent behavior can be produced. In working with ACT-R, we found that it was a good instantiation of working memory but that we needed to add other aspects of cognition, including LTM and perception, to have a complete cognitive system. We note progress to date and address the remaining challenges.

15. SUBJECT TERMS
cognitive architectures, robotics, cognition, working memory, context

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Troy Dale Kelley
Unclassified	Unclassified	Unclassified	UU	20	19b. TELEPHONE NUMBER (Include area code) (410) 278-5869

Contents

List of Figures

1. Introduction

The Symbolic and Subsymbolic Robotics Intelligence Control System (SS-RICS) was developed by the US Army Research Laboratory's Human Research and Engineering Directorate in cooperation with Towson University beginning in 2004. The goal of the program was to develop a system capable of performing a wide variety of autonomous behaviors under a variety of battlefield conditions.

As a general theoretical position, we have taken the stance that cognition arises from a collection of different algorithms, each with different functionalities that together produce the integrated process of cognition. This is also known as a functionalist representation.[1] We are developing SS-RICS to be a modular system or a collection of modular algorithms, each group of algorithms with different responsibilities for the functioning of the overall system. The important component is the interaction or interplay among these different algorithms, which leads to an integrated cognitive system. We are not necessarily attempting to produce a neurological representation of the individual components of the brain (thalamus, amygdale) but instead a functional representation of cognition (learning, memory).

We began the development of SS-RICS by using the existing cognitive architecture the Adaptive Control of Thought – Rational (ACT-R)[2] as a framework. ACT-R has a long history of development and has continued to be refined to the present day. ACT-R grew from the artificial intelligence symbolic tradition beginning with Newell and Simon[3] and their work on the generalized problem solver (GPS) and their studies of the problem-solving strategies of chess masters. ACT-R grew from the production system-based work of GPS and was later augmented to include memory algorithms developed by John Anderson, and subsequently to include a variety of additional learning methods and algorithms. ACT-R has been used primarily to simulate human performance data and to make predictions of human performance and error data.[4] ACT-R currently enjoys a large user base and undergoes continuous revisions and improvement.

In our work on the development of SS-RICS, we found that ACT-R was a sufficient approximation of working memory for a robotic system but that the robot needed other systems to function in a dynamic world, primarily perceptual systems and long-term memory (LTM). The perceptual systems within ACT-R were not sufficient for robotic control, and the LTM mechanisms were model specific and not generalized knowledge. Consequently, we added LTM to SS-RICS by using ConceptNet[5], and we added a variety of perceptual systems to SS-RICS by using statistical techniques and neural networks.

2. System Overview

2.1 System Architecture

The primary architectural mechanisms that make up SS-RICS can be broken down into 3 main sections: sensory input, subsymbolic processing, and symbolic processing. Figure 1 depicts these sections as the large boxes, each of which contains collections of other subsystems.

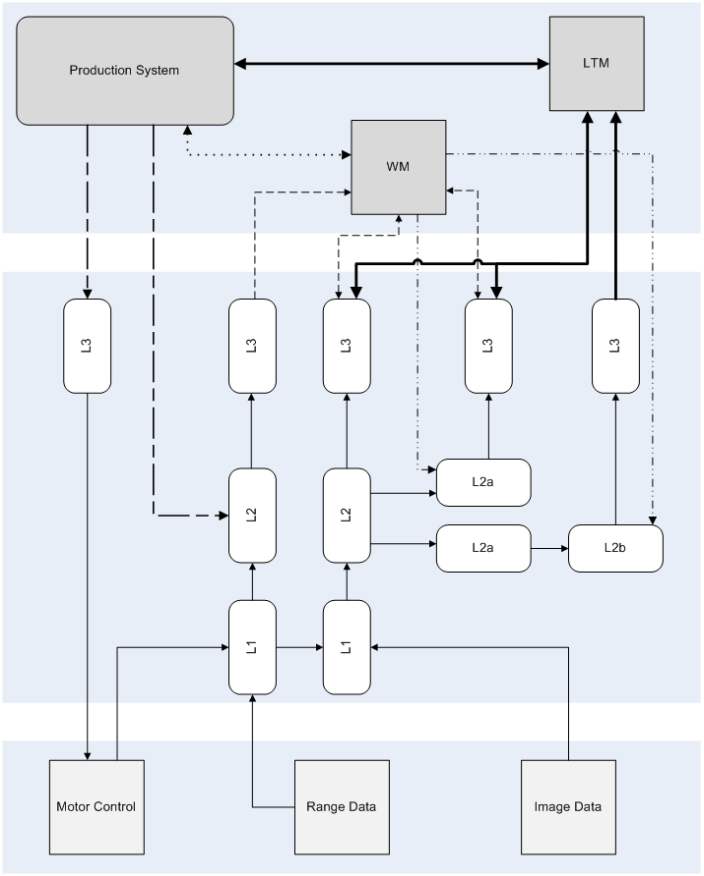

Fig. 1 An overview of the main section of the SS-RICS architecture: level 1 (L1) processors, level 2 (L2) processors, and level 3 (L3) processors

Starting from the bottom of Fig. 1, the lowest level is focused on interfacing and collecting sensory input from input/output (I/O) systems, such as motor control, range, and image sensors. At this level we are dealing with abstract sensor models that encapsulate device driver interfaces to allow for modularity. Motor, range, and image are by no means the only sensory I/O systems that can be handled by SS-RICS, but we will discuss these to simplify things. As the sensory data are collected from these systems, they are handed off to the low-level processing made up of what we have termed subsymbolic processors.

The subsymbolic processors are composed of algorithms that include artificial neural networks, collections of busted classifiers, and brute force data processing. The processors can be a single serial operation or a pipeline of

several operations containing both serial and parallel components. There are 3 main levels of subsymbolic processors. Level 1 (L1) is concerned with capturing sensory data packaging and transforming it into common internal data structures. Additionally, L1 processors may also fuse data with other L1 processor output. Level 2 (L2) processors are concerned primarily with merging internal working memory structures with incoming sensory data that has been collected by L1. L2 may also contain several sublevels that deal with fusing data and updating internal data structures, such as those used for low-level localization and spatial interaction.

At the highest level we have the production system. The production system is composed of one or more threads that may interact with both working memory and LTM. Working memory is the result of low-level sensory processing and activation of LTM by both the subsymbolic processors and production system processing. As can be seen in Fig. 1, the production system may interact with L2 and L3 subsymbolic processors, allowing higher-level decisions to guide processing of sensory data and manipulate lower-level motor systems.

The production system executes goals that can be considered simple operations composed of several rules. Each rule can be conceptualized as an if–then type operation. That is, a set of antecedents are matched against working memory using Boolean and–or operators to result in a TRUE or FALSE value. When all antecedents are evaluated to TRUE, the rule is allowed to fire. Upon firing, a collection of consequents are executed that may interact with working memory by adding, removing, or updating data as well as retrieving memory from long-term storage. Goals may also be combined with other goals in serial or parallel fashion. For example, the goal to move to a door may have several subgoals: first to locate the door, then to plan a path to the door, and finally to move to the door. If we were to add the task of finding an object while on route to the door, we could have a parallel goal executed: to look for the object.

2.2 Goals and Rules

The syntax used for goal and rule creation is similar to functional block definitions found in C and C++. A goal definition starts with the block descriptor “Goal” followed by the name of the goal; within the goal block we then define any number of rules. A rule is defined starting with the “Rule” descriptor followed by its name. We then define zero or more antecedents in the form (Name Type ID = Value…), which will be compared to working memory. The antecedents are then connected with the && and || logical operators, generating a single Boolean result. Within the rule block we define a set of consequents that manipulate working memory and interact with the world. Here is an example of a simple goal to meet a person who is either known or unknown to the system:

```
Goal MeetSomeone
{
        Rule UnknownPerson (MeetSomeone Arg Value=True) &&
                           (PersonName Arg Value!=NULL) &&
                           NOT(?PersonName.Value Person Meet=True)
        {
                Execs(Voice Say “Hello ?PersonName.Value_ it is very nice to meet you.”)
                Set(MeetSomone Arg Value=False)
                Set(?PersonName.Value Person Meet=True)
        }
        Rule KnownPerson (MeetSomeone Arg Value=True) &&
                           (PersonName Arg Value!=NULL) &&
                           (?PersonName.Value Person Meet=True)
        {
                Execs(Voice Say “Hello ?PersonName.Value_ it’s nice to see you again.”)
                Set(MeetSomone Arg Value=False)
        }
}
```

In this example, we have defined one goal, MeetSomeone, composed of 2 rules, UnknownPerson and KnownPerson. The UnknownPerson rule will fire when we have a match to 3 facts in working memory. First, we must have a fact in working memory named MeetSomeone with a type Arg and a slot named Value with a value of True. Next, we must have a fact in working memory named PersonName with a non-NULL Value slot. Last, we must not have a fact in working memory with a name equal to the name of the PersonName Value slot. When the UnknownPerson rule fires, it will execute 3 consequents. First, it tells the Voice subsymbolic processor to execute the Say command synchronously and greet the person with the phrase “Hello [Person’s Name] it is very nice to meet you.” Second, the MeetSomeone memory is updated, changing the Value slot of the MeetSomeone memory to False. Last, a memory is added to working memory with the name of the person that was just met. The KnownPerson rule has essentially the same collection of antecedents. However, the last antecedent requires a memory in working memory named with the person we are meeting that would have been added by the NewPerson rule. The rule then executes a Say command similar to the UnknownPerson rule; however, it greets the person with “Hello [Person’s Name] it’s nice to see you again.”

2.3 Long-Term Memory (LTM)

Typically in ACT-R, a modeler would develop any LTM as part of the development of the model. For the development of SS-RICS, we thought this might place too much of a burden on the modeler. We wanted a predefined semantic network or relational database that would be a storehouse LTM and allow SS-RICS to reason about the world.

ConceptNet is a freely available semantic network from the Massachusetts Institute of Technology. It contains 1.6 million assertions. These assertions can be used to query the database for knowledge on a variety of subjects.

Figure 2 shows some of the typical assertions found in ConceptNet. The *f* counts are the number of times the fact is found in the original Open Mind Common Sense corpus. The *i* counts show how many times the assertion was inferred during a "relaxation" process. The relaxation process is meant to improve connectivity within the network and smooth over semantic gaps.

Sample Concept Net Assertions

K-LINES (1.25 million assertions) (ConceptuallyRelatedTo "bad breath" "mint" "f=4;i=0;") (ThematicKLine "wedding dress" "veil" "f=9;i=0;") (SuperThematicKLine "western civilization" "civilization" "f=0;i=12;")

THINGS (52,000 assertions) (IsA "horse" "mammal" "f=17;i=3;") (PropertyOf "fire" "dangerous" "f=17;i=1;") (PartOf "butterfly" "wing" "f=5;i=1;")

(MadeOf "bacon" "pig" "f=3;i=0;") (DefinedAs "meat" "flesh of animal" "f=2;i=1;")

AGENTS (104,000 assertions) (CapableOf "dentist" "pull tooth" "f=4;i=0;")

EVENTS (38,000 assertions) (PrerequisiteEventOf "read letter" "open envelope" "f=2;i=0;") (FirstSubeventOf "start fire" "light match" "f=2;i=3;") (SubeventOf "play sport" "score goal" "f=2;i=0;") (LastSubeventOf "attend classical concert" "applaud" "f=2;i=1;")

Fig. 2 ConceptNet assertions

We were primarily interested in using ConceptNet to help us with contextual relationships and to aid in generalized perception. For example, if the robot knew, or was told, that it was in a building, then it could query ConceptNet to find out that buildings contained doors and hallways. This could then be used as a contextual mechanism so that if the robot saw an object, it would assume that it was probably a door, and not a tree, for instance.

In using ConceptNet, we found that it was fast enough for real-time assistance with contextual problems. Most queries were on the order of just a few seconds per query. For queries where a lot of information was required, a caching algorithm was used to speed up the process.

One problem we found with ConceptNet was that it did not contain the depth and breadth of information we needed to improve contextual-based queries for object recognition tasks. For example, to improve our robot's perceptual capabilities in indoor environments, we needed detailed information from ConceptNet about doors, such as their size, shape, and possibly their texture, or the fact that doorknobs are frequently found on doors. For this type of query, ConceptNet did not have enough detailed information about typical indoor objects. However, this type of information could be added, and we are currently investigating ways in which we can expand our implementation of ConceptNet in real-time to include information that has been learned by SS-RICS.

Additionally, we found unusual instances where queries that seemed to be identical were different from each other. For example, a query for "Dog" with a capital "D" produced different results than "dog" with a lowercase "d".

Presumably, this might be because a word beginning with a capital letter could connote a proper name. We found this to be troublesome in the long run.

Eventually, we aim to find that ConceptNet will provide SS-RICS with a wide range of knowledge about the world. Our goal is to use this knowledge not only for object recognition but also for logical inference and deductions about the world.

2.4 Perceptual Capabilities

Within SS-RICS we utilize a variety of different perceptual algorithms. In a broad sense, we refer to these algorithms as the subsymbolic systems because during initial development, we concentrated heavily on using neural networks for perceptual processing (neural networks are frequently referred to as subsymbolic systems). As we have further developed SS-RICS, we have decided not to limit ourselves to strictly subsymbolic systems for perceptual capabilities; we have looked at using other algorithms that are not typically thought of as subsymbolic techniques, including AdaBoost and K-means.

In general, our subsymbolic systems run in parallel and produce information that can be used by the production system. For example, if the subsymbolic system identifies a door, it will create an instance of the door and feed that information as a fact, or declarative memory type, to the production system. The production system might then generate an action such as “go to door1”. Currently, we do not use the same buffer system that is being used by ACT-R to limit the amount of information that gets fed into the production system from the outside world. We are still trying to determine which, if any, perceptual system (visual, auditory) requires buffers or whether one buffer for all the perceptual systems will suffice.

Additionally, each one of our subsymbolic systems runs as its own separate thread within the overall architecture. This is so that an individual subsymbolic system can be shut down if SS-RICS needs more resources. This structure helps us manage resources and allows the system to reallocate resources as needed, depending on the tasks at hand.

We understood from the beginning that perception is a difficult task but felt that even limited perception would be useful to a robotic system and to the performance of the overall system. If the perception was not perfect, perhaps this could be overcome by learning or interactions with the operator.

2.4.1 Geon-Based Recognition

We have been working to develop the Cognitive Object Recognition System (CORS) in cooperation with UtopiaCompression. The CORS was inspired by current psychophysical research in which multiple recognition algorithms are

integrated to provide a comprehensive solution for object recognition and landmark recognition. Geon-based recognition[6] (Fig. 3) uses predefined shapes (cubes, tubes) to match to objects in the environment. CORS uses Geon-based recognition as well as feature-based recognition to provide 2 types of object recognition capabilities for SS-RICS. We are hoping the Geon-based system will be useful for the recognition of man-made objects (tables, chairs), and the feature-based system will be useful for the recognition of environmental objects (clouds, trees). This work is ongoing. While this work was beginning, we were able to develop our own neural networks for perceptual processing within SS-RICS.

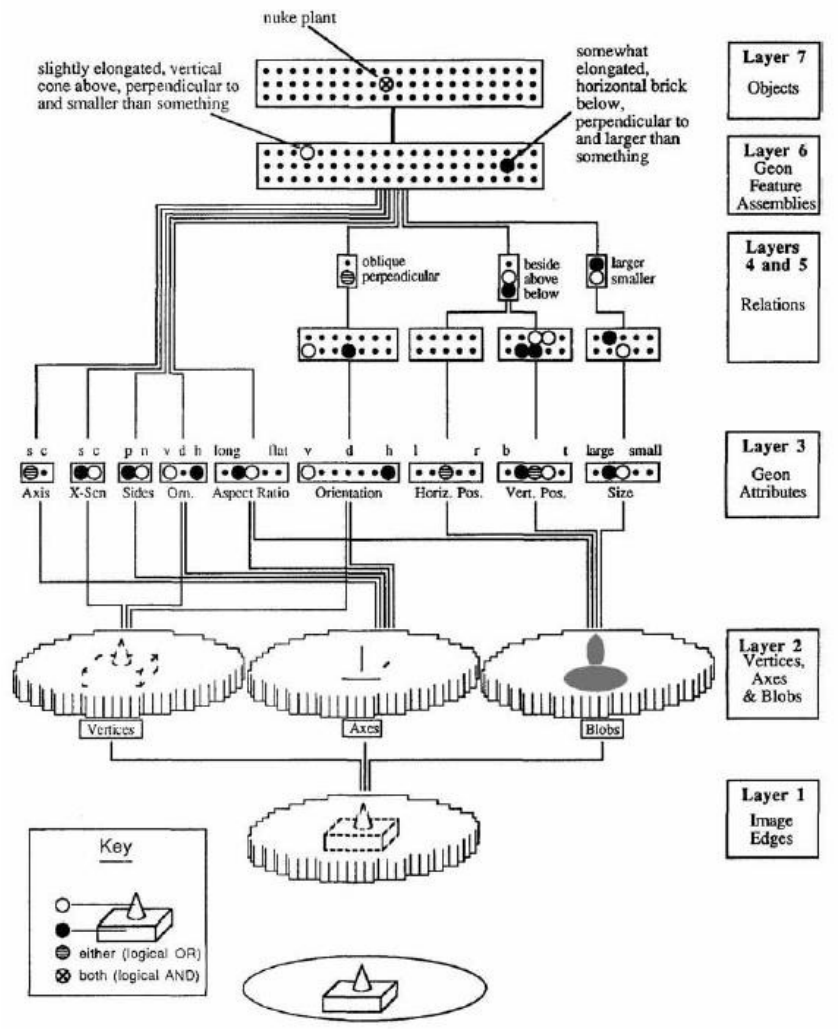

Fig. 3 Geon Object Recognition System[6]

2.4.2 Neural Networks

We initially began working with traditional feed-forward neural networks using lidar data. We used a lidar laser, which scans the horizon in 180° increments. This scan produces an aerial view of the environment as a horizontal plane. It is not a feature-rich view of the world, but it is a good place to start because of the simplicity of the data. Additionally, each scan of the world produces a vector, which can then be easily fed into a neural network.

Our work concentrated on using laser scans to determine hallway intersection types. For example, we trained a network to recognize right-hand turns, left-hand turns, T-intersections, and straight intersections. The goal was to then use these intersections in a topological map of the environment so that the robot could navigate through the world. The map would link the intersections together and allow for navigation in hallways. We were able to use this methodology in a maze environment at the National Institute of Standards and Technology (NIST) test facility.

However, we did eventually find ourselves faced with a common problem associated with neural networks, which is the catastrophic forgetting problem.[7] The catastrophic forgetting problem is one where a network cannot be trained on new information without also being retrained on all of the previous information that it has learned. For example, suppose we trained our network on left- and right-hand turns and found ourselves in a hallway with a dead end, but we did not train our network on this ahead of time. If we wanted to add the dead end as a category, we would have to retrain the network on all of the existing data, the right turns and left turns, as well as the dead-end category in order to develop a new network. This problem prohibits neural networks from being used as an adaptive mechanism that can be retrained on the fly to learn newly encountered information. However, we did find that as long as the problem set was well defined, we could use traditional neural networks to develop a set of primitives to recognize and utilize other algorithms for adaptive learning.

2.4.3 Adaptive Resonance Theory (ART) Networks

The catastrophic forgetting problem of traditional neural networks led us to investigate Adaptive Resonance Theory (ART)[8] networks as a solution to the problem. ART networks are designed so that they can be retrained with new information and do not suffer from the catastrophic forgetting problem. Specifically, we used ARTMAP, which is a supervised learning methodology, to train on a variety of different objects. We found the ART networks to be very useful for the classification of objects, given a clean set of data, and especially if we could input a vector into the network—more importantly, a vector that was representative of the object being learned. This was relatively easy when

we were using lidar laser data since the data were fairly clean and the important aspects of the representation were easily captured as a vector. However, this was more difficult with image processing because the image is typically represented as a matrix instead of a vector. This led us to the investigation of Principal Components Analysis (PCA).

2.4.4 Principal Components Analysis (PCA)

PCA is a statistical technique for the analysis of data which allows for the identification of "principal factors" that represent the overall data. These principal factors are reduced from the data and are uncorrelated with each other. The procedure is a statistical way of describing variance in data and identifying the "principal components" of the variance. While this might be useful for statistical data analysis, we found it to be marginally useful for image processing.

PCA was not developed originally for image processing; instead, it was developed for exploratory data analysis. Translating an image matrix into a vector to identify the "first principal component" was not the intended purpose of the algorithm. Subsequently, this creates problems when trying to use the algorithm for image processing. The problem lies in the selection of the first principal component for the vector. For example, given an image of a soda can with a large glare or shadow on the can, the algorithm might identify the shadow or glare as the most important, or first principal component, of the image. Even if these spurious features are preprocessed and removed from the final image, the algorithm might again identify something meaningless in the image as the first principal component.

For example, when teaching a child how to identify the difference between a cowboy hat and a baseball hat, one might say, "Look at the brim of the cowboy hat. See how the brim goes all the way around the hat? Now, look at the baseball hat. The brim is only in the front of the hat, and it is long and narrow. That is how you tell the difference." In this example, the brim is the most important aspect of distinguishing between a cowboy hat and a baseball hat. But if we were to subject pictures of hats to PCA, it may or may not identify the brim of the hat as the first principal component; it might identify something else as the first principal component.

While we were pleased with the ability of PCA to convert matrix data into a vector, which could then be fed into our ART networks, we were not satisfied with the selection of the principal components of the image. This led us to become very aware of the differences between instance-based recognition and generalized recognition as a problem.

2.4.5 Instance-Based and Generalized Recognition

Our work with PCA, and to a limited degree other similar techniques (Scale Invariant Feature Transform [SIFT]), led us to the realization that instance-based recognition was relatively easy for computer systems, but generalized recognition was going to be more difficult. For example, using either PCA or SIFT, we could recognize a specific instance of an object. Both of these techniques extract essential features of an object, and in the case of SIFT, allow the object to be transformed (rotated) and still recognized. We also worked with simple image correlation algorithms and found them to be useful for the recognition of a particular object or the instance of a particular object. Additionally, we used image correlation to create a small sample of the image, then we were able to search for that sample within a larger image to find a match in the overall image. We are working to leverage easy recognition for instances of objects so that SS-RICS generalization capabilities are increased to allow the system to recognize general classes of objects or recognize objects that have never been seen by the system before. It could be that the instances provide the data from which generalizations can be drawn and used for the recognition of new objects, but we are still working on this idea.

3. Conclusions

This report details the current development of SS-RICS. We would like to be clear about what has been implemented within SS-RICS and what we are still developing. Currently, SS-RICS contains a stable production system capable of executing commands such as “Go to Door”. The system can use a predefined map to generate a path to a door or any other object in the environment. Additionally, the operator can label the door as “Bob’s Door”, for example, and give the command, “Go to Bob’s Door”. The system uses off-the-shelf voice recognition technology so that the operator can vocalize commands to control the system. The production system itself is very stable and has been used extensively to develop a variety of goal-based behaviors, including navigation and problem-solving tasks. For example, we used the production system to solve a maze at NIST. The goals developed to solve the maze were relatively few (15 total) and worked reasonably well.

The subsymbolic systems within SS-RICS are stable and run in parallel. The ability to add additional functionality to the subsymbolic systems has been done within SS-RICS and is relatively easy to accomplish. We recently added Decision Field Theory[9] to the production system, which allowed SS-RICS to make choices about possible paths in its environment. We have not experimented with shutting down subsymbolic systems to reallocate resources; we hope to do some of this work soon.

The traditional feed-forward neural networks and the ART neural networks have been implemented and tested. We have also implemented pulsed neural networks, which were not discussed in this report because we have not been testing them extensively but hope to test them in the future.

The implementation of ConceptNet is still in its infancy and has just started. We are currently able to access the entire ontology and pull in bits of information into SS-RICS as working memory elements. We hope to use ConceptNet for logical inference and problem solving in the future.

The CORS is still in development as a Small Business Innovative Research contract by UtopiaCompression. The contract has just entered its Phase II development cycle.

4. References

1. Fodor JA. Representations: philosophical essays on the foundations of cognitive science. Cambridge (MA): MIT Press; 1981. ISBN 0-262-06079-5.

2. Anderson JR, Lebiere C. The atomic components of thought. Mahwah (NJ): Lawrence Erlbaum Associates; 1998.

3. Newell A, Simon HA. GPS, a program that simulates human thought. In: Feigenbaum EA, Feldman J, editors. Computers and thought. New York (NY): McGraw-Hill; 1963. p 279–293.

4. Kelley TD, Patton DJ, Allender L. Predicting situation awareness errors using cognitive modeling. In: MJ Smith, RJ Koubek, G Salvendy, D Harris, editors. Proceedings of HCI International 2001. Usability evaluation and interface design: cognitive engineering, intelligent agents and virtual reality; Vol. 1. 2001 Aug; New Orleans, LA. Mahwah (NJ); Lawrence Erlbaum Associates; 2001. pp. 1455–1459.

5. Liu H, Singh P. ConceptNet—a practical commonsense reasoning tool-kit. BT Technology Journal. 2004;22(4):211–226.

6. Hummel JE, Biederman I. Dynamic binding in a neural network for shape recognition. Psychological Review. 1992;99:480–517.

7. French RM. Catastrophic forgetting in connectionist networks. Trends in Cognitive Sciences. 1 April 1999;3(4):128–135.

8. Carpenter GA, Grossberg S, Reynolds JH. ARTMAP: supervised real-time learning and classification of nonstationary data by a self-organizing neural network. Neural Networks. 1991;4:565–588.

9. Busemeyer JR, Townsend JT. Decision field theory: a dynamic cognition approach to decision making. Psychological Review. 1993;100:432–459.

List of Symbols, Abbreviations, and Acronyms

ACT-R	Adaptive Control of Thought - Rational
ART	Adaptive Resonance Theory
CORS	Cognitive Object Recognition System
GPS	generalized problem solver
I/O	input/output
LTM	long-term memory
NIST	National Institute of Standards and Technology
PCA	Principal Components Analysis
SIFT	Scale Invariant Feature Transform
SS-RICS	Symbolic and Subsymbolic Robotics Intelligence Control System

1 (PDF) DEFENSE TECHNICAL INFORMATION CTR DTIC OCA

2 (PDF) DIR ARL
IMAL HRA
RECORDS MGMT
RDRL DCL
TECH LIB

1 (PDF) GOVT PRINTG OFC
A MALHOTRA

1 (PDF) ARL
RDRL HRB B
T DAVIS
BLDG 5400 RM C242
REDSTONE ARSENAL AL
35898-7290

8 (PDF) ARL
SFC PAUL RAY SMITH CENTER
RDRL HRO COL H BUHL
RDRL HRF J CHEN
RDRL HRA I MARTINEZ
RDRL HRR R SOTTILARE
RDRL HRA C
A RODRIGUEZ
RDRL HRA B G GOODWIN
RDRL HRA A C METEVIER
RDRL HRA D B PETTIT
12423 RESEARCH PARKWAY
ORLANDO FL 32826

1 (PDF) USA ARMY G1
DAPE HSI J WARNER
300 ARMY PENTAGON
RM 2C489
WASHINGTON DC
20310-0300

1 (PDF) USAF 711 HPW
711 HPW/RH K GEISS
2698 G ST BLDG 190
WRIGHT PATTERSON AFB OH
45433-7604

1 (PDF) USN ONR
ONR CODE 341
J TANGNEY
875 N RANDOLPH STREET
BLDG 87
ARLINGTON VA 22203-1986

1 (PDF) USA NSRDEC
RDNS D D TAMILIO
10 GENERAL GREENE AVE
NATICK MA 01760-2642

1 (PDF) OSD OUSD ATL
HPT&B B PETRO
4800 MARK CENTER DRIVE
SUITE 17E08
ALEXANDRIA VA 22350

ABERDEEN PROVING GROUND

11 (PDF) ARL
RDRL HR
J LOCKETT
P FRANASZCZUK
K MCDOWELL
K OIE
RDRL HRB
D HEADLEY
RDRL HRB C
J GRYNOVICKI
RDRL HRB D
C PAULILLO
RDRL HRF A
A DECOSTANZA
RDRL HRF B
A EVANS
RDRL HRF C
J GASTON
RDRL HRF D
A MARATHE
T KELLEY