DTIC FILE COPY

Volume II

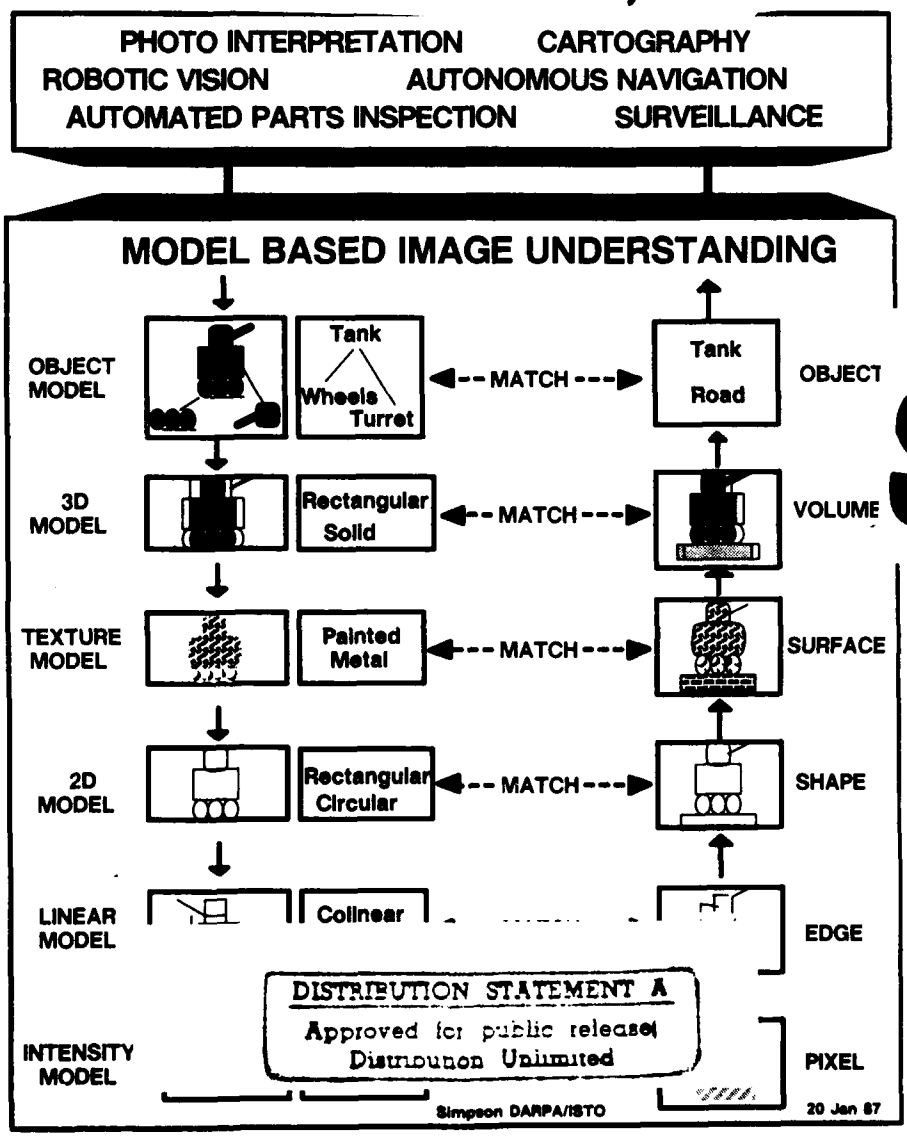
PROCEEDINGS:

Image Understanding Workshop

(1)

AD-A221 428

87-0209



MR-
DTIC

DTIC
ELECTE
MAY 15 1990
S D
D

Sponsored by:
Defense Advanced Research Projects Agency
Information Science and Technology Office

DARPA

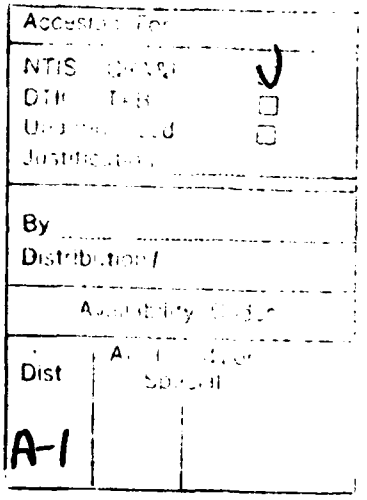
February 1987

DISTRIBUTION STATEMENT A
Approved for public release
Distribution Unlimited

Image Understanding Workshop

Proceedings of a Workshop
Held at
Los Angeles, California

February 23-25, 1987



Volume II

Sponsored by:

Defense Advanced Research Projects Agency
Information Science and Technology Office

This document contains copies of reports prepared for the DARPA Image Understanding Workshop. Included are results from both the basic and strategic computing programs within DARPA/ISTO sponsored projects.

APPROVED FOR PUBLIC RELEASE
DISTRIBUTION UNLIMITED

The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied of the Defense Advanced Research Projects Agency or the United States Government.

90 05 14 102

Distributed by
Morgan Kaufmann Publishers, Inc.
95 First Street
Los Altos, California 94022
ISBN 0-934613-36-2
Printed in the United States of America

TABLE OF CONTENTS

Page

AUTHOR INDEX i

VOLUME I

SECTION I - PROGRAM REVIEWS BY PRINCIPAL INVESTIGATORS

"Vision in Dynamic Environments", Azriel Rosenfeld, Larry S. Davis, John (Yiannis) Aloimonos; University of Maryland 1

"USC Image Understanding Research - 1986", R. Nevatia; University of Southern California 8

"Image Understanding Research at SRI International", Martin A. Fischler and Robert C. Bolles; SRI International 12

"Image Understanding: Intelligent Systems", Thomas O. Binford; Stanford University 18

"Image Understanding Research at CMU", Takeo Kanade; Carnegie-Mellon University 32

"MIT Progress in Understanding Images", T. Poggio and the Staff; Massachusetts Institute of Technology 41

"Summary of Progress in Image Understanding at the University of Massachusetts", Allen R. Hanson and Edward M. Riseman; University of Massachusetts at Amherst 55

"Recent Progress of the Rochester Image Understanding Project", Jerome A. Feldman and Christopher M. Brown; University of Rochester 65

"Image Understanding and Robotics Research at Columbia University", John R. Kender, Peter K. Allen, Terrance E. Boult; Columbia University 71

TABLE OF CONTENTS

Page

SECTION I - PROGRAM REVIEWS BY PRINCIPAL INVESTIGATORS (Continued)

"Developments in Knowledge-Based Vision for Obstacle Detection and Avoidance", K.E. Olin, F.M. Vilnrotter, M.J. Daily and K. Reiser; Hughes Research Laboratories .. 78

"Detecting Obstacles in Range Imagery", by Michael J. Daily, John G. Harris, and Kurt Reiser; Hughes Artificial Intelligence Center 87

"Model-Directed Object Recognition on the Connection Machine", D.W. Thompson and J.L. Mundy; General Electric Corporate Research and Development 98

"Environmental Modeling and Recognition for an Autonomous Land Vehicle", Daryl T. Lawton, Tod S. Levitt, Christopher C. McConnell, Philip C. Nelson, Jay Glicksman; Advanced Decision Systems 107

"Honeywell Progress on Knowledge-Based Robust Target Recognition & Tracking", Bir Bhanu, Durga Panda and Raj Aggarwal; Honeywell Systems & Research Center 122

SECTION II - TECHNICAL REPORTS PRESENTED

"The ITA Range Image Processing System", David G. Morgenthaler, Keith D. Gremban, Mitch Nathan, John D. Bradstreet; Martin Marietta Denver Aerospace 127

"Vision and Navigation for the Carnegie Mellon Navlab", Charles Thorpe, Steven Shafer, Takeo Kanade and the members of the Strategic Computing Vision Lab, Carnegie-Mellon University 143

TABLE OF CONTENTS

Page

SECTION II - TECHNICAL REPORTS PRESENTED (Continued)

"Vision-Based Navigation: A Status Report", Larry S. Davis, Daniel Dementhon, Ramesh Gajulapalli, Todd R. Kushner, Jacqueline Le Moigne, Phillip Veatch; University of Maryland 153

"Information Management in a Sensor-based Autonomous System", Grahame B. Smith and Thomas M. Strat; SRI International 170

"Tools and Experiments in the Knowledge-Directed Interpretation of Road Scenes", Bruce A. Draper, Robert T. Collins, John Brolio, Joey Griffith, Allen R. Hanson, Edward M. Riseman; University of Massachusetts at Amherst 178

"Models of Errors and Mistakes in Machine Perception - Part 1. First Results for Computer Vision Range Measurements", Ruzena Bajcsy, Eric Krotkov, Max Mintz; University of Pennsylvania 194

"Automating Knowledge Acquisition for Aerial Image Interpretation", David M. McKeown, Jr., Wilson A. Harvey; Carnegie-Mellon University 205

"Using Generic Geometric Models for Intelligent Shape Extraction", Pascal Fua and Andrew J. Hanson; SRI International 227

"Stereo Correspondence: A Hierarchical Approach", Hong Seh Lim and Thomas O. Binford; Stanford University 234

"Symbolic Pixel Labeling for Curvilinear Feature Detection", John Canning, J. John Kim, Azriel Rosenfeld; University of Maryland 242

"Searching for Geometric Structure in Images of Natural Scenes", George Reynolds, R. Ross Beveridge; University of Massachusetts at Amherst 257

TABLE OF CONTENTS

Page

SECTION II - TECHNICAL REPORTS PRESENTED (Continued)

"Detecting Runways in Aerial Images", A. Huertas, W. Cole and R. Nevatia; University of Southern California 272

"A Report on the DARPA Image Understanding Architectures Workshop", Azriel Rosenfeld; University of Maryland 298

"Image Processing to Geometric Reasoning: Military Image Analysis at GE FESD", M.S. Horwedel; General Electric Federal Electronic Systems Division 303

"Image Understanding Technology and its Transition to Military Applications", David Y. Tseng; Hughes Research Laboratories; Julius F. Bogdanowicz; Electo-Optical and Data Systems Group, Hughes Aircraft Company 310

"Context Dependent Target Recognition", Teresa M. Silberberg; Hughes Artificial Intelligence Center 313

"Precompiling a Geometrical Model into an Interpretation Tree for Object Recognition in Bin-picking Tasks", Katsushi Ikeuchi; Carnegie-Mellon University 321

"Analytical Properties of Generalized Cylinders and Their Projections", Jean Ponce, David Chelberg, Wallace Mann; Stanford University 340

"Surface Segmentation and Description from Curvature Features", T.J. Fan, G. Medioni, and R. Nevatia; University of Southern California 351

"From Sparse 3-D Data Directly to Volumetric Shape Descriptions", Kashipati Rao and R. Nevatia; University of Southern California 360

TABLE OF CONTENTS

Page

SECTION II - TECHNICAL REPORTS PRESENTED (Continued)

"Object Recognition Using Alignment", Daniel P. Huttenlocher, Shimon Ullman; Massachusetts Institute of Technology 370

"Parallel Recognition of Objects Comprised of Pure Structure", Paul R. Cooper and Susan C. Hollbach; University of Rochester ... 381

"A Framework for Implementing Multi-Sensor Robotic Tasks", Peter K. Allen; Columbia University 392

"Shape from Darkness: Deriving Surface Information from Dynamic Shadows", John R. Kender, Earl M. Smith; Columbia University ... 399

VOLUME II

SECTION III - OTHER TECHNICAL REPORTS

"Vision and Visual Exploration for the Stanford Mobile Robot", Ernst Triendl and David J. Kriegman; Stanford University 407

"AuRA: An Archiecture for Vision-Based Robot Navigation", Ronald C. Arkin, Edward M. Riseman, Allan R. Hanson; University of Massachusetts at Amherst 417

"Guiding an Autonomous Land Vehicle Using Knowledge-Based Landmark Recognition", Hatem Nasr, Bir Bhanu and Stephanie Schaffer; Honeywell Systems and Research Center 432

"The CMU Navigational Architecture", Anthony Stentz, Yoshimasa Goto; Carnegie-Mellon University 440

"Qualitative Navigation", Tod S. Levitt, Daryl T. Lawton, David M. Chelberg, Philip C. Nelson; Advanced Decision Systems 447

TABLE OF CONTENTS

Page

SECTION III - OTHER TECHNICAL REPORTS (Continued)

"Interpretation of Terrain Using Hierarchical Symbolic Grouping for Multi-Spectral Images", Bir Bhanu and Peter Symosek; Honeywell Systems & Research Center 466

"Design of a Prototype Interactive Cartographic Display and Analysis Environment", Andrew J. Hanson, Alex P. Pentland, and Lynn H. Quam; SRI International 475

"The Image Understanding Architecture", Charles C. Weems, Steven P. Levitan; University of Massachusetts at Amherst 483

"Constructs for Cooperative Image Understanding Environments", Christopher C. McConnell, Philip C. Nelson, Daryl T. Lawton; Advanced Decision Systems 497

"Interpreting Aerial Photographs by Segmentation and Search", David Harwood, Susan Chang, Larry S. Davis; University of Maryland 507

"Initial Hypothesis Formation in Image Understanding Using an Automatically Generated Knowledge Base", Nancy Bonar Lehrer, George Reynolds, Joey Griffith; University of Massachusetts at Amherst 521

"Goal-Directed Control of Low-Level Processes for Image Interpretation", Charles A. Kohl, Allen R. Hanson, Edward M. Riseman; University of Massachusetts at Amherst 538

"Active Vision", John (Yiannis) Aloimonos, Isaac Weiss; University of Maryland; Amit Bandyopadhyay; SUNY Stony Brook 552

TABLE OF CONTENTS

Page

SECTION III - OTHER TECHNICAL REPORTS (Continued)

"An Integrated System That Unifies Multiple Shape From Texture Algorithms", Mark L. Moerdler and John R. Kender; Columbia University 574

"DRIVE - Dynamic Reasoning from Integrated Visual Evidence", Bir Bhanu and Wilhelm Burger; Honeywell Systems & Research Center 581

"What Is A "Degenerate" View?", John R. Kender, David G. Freudenstein; Columbia University 589

"The Role and Use of Color in a General Vision System", Glenn Healey and Thomas O. Binford; Stanford University ... 599

"Using a Color Reflection Model to Separate Highlights from Object Color", Gudrun J. Klinker, Steven A. Shafer, and Takeo Kanade; Carnegie-Mellon University 614

"Recognizing Unexpected Objects: A Proposed Approach", Azriel Rosenfeld; University of Maryland 620

"Parallel Algorithms for Computer Vision on the Connection Machine", James J. Little, Guy Blelloch, and Todd Cass; Massachusetts Institute of Technology 628

"Solving The Depth Interpolation Problem on a Fine Grained, Mesh- and Tree-Connected SIMD Machine", Dong J. Choi; Columbia University 639

"Survey of Parallel Computers", Hong Seh Lim and Thomas O. Binford; Stanford University 644

TABLE OF CONTENTS

Page

SECTION III - OTHER TECHNICAL REPORTS (Continued)

"Evidence Combination Using Likelihood Generators", David Sher; The University of Rochester 655

"Multi-Modal Segmentation using Markov Random Fields", Paul B. Chou and Christopher M. Brown; The University of Rochester 663

"Minimization of the Quantization Error in Camera Calibration", Behrooz Kamgar-Parsi; University of Maryland 671

"Uncertainty Analysis of Image Measurements", M.A. Snyder; University of Massachusetts at Amherst 681

"Results of Motion Estimation With More Than Two Frames", Hormoz Shariat and Keith E. Price; University of Southern California 694

"Tracing Finite Motions Without Correspondence", Ken-ichi Kanatani and Tsai-Chia Chou; University of Maryland 704

"A Unified Perspective on Computational Techniques for the Measurement of Visual Motion", P. Anandan; University of Massachusetts at Amherst 719

"Hierarchical Gradient-Based Motion Detection", Frank Glazer; University of Massachusetts at Amherst 733

"Shape Reconstruction on a Varying Mesh", Isaac Weiss; University of Maryland 749

"Stereo Matching Using Viterbi Algorithm", G.V.S. Raju; Ohio University, Thomas O. Binford; Stanford University, and S. Shekhar; Stanford University 766

TABLE OF CONTENTS

Page

SECTION III - OTHER TECHNICAL REPORTS (Continued)

"Steps Toward Accurate Stereo Correspondence", Steven D. Cochran; University of Southern California 777

"Stereo Matching by Hierarchical, Microcanonical Annealing", Stephen T. Barnard; SRI International 792

"The Formation of Partial 3D Models from 2D Projections - An Application of Algebraic Reasoning", D. Cyrluk, D. Kapur, J.L. Mundy, and V. Nguyen; General Electric Corporate Research and Development .. 798

"Spectral and Polarization Stereo Methods Using a Single Light Source", Lawrence Brill Wolff; Columbia University 810

"Surface Curvature and Contour from Photometric Stereo", Lawrence Brill Wolff; Columbia University 821

"Qualitative Information in the Optical Flow", Alessandro Verri and Tomaso Poggio; Massachusetts Institute of Technology 825

"Algorithm Synthesis for IU Applications", Michael R. Lowry; Stanford University 835

"Generalizing Epipolar-Plane Image Analysis for Non-Orthogonal and Varying View Directions", H. Harlyn Baker, Robert C. Bolles, and David H. Marimont; SRI International 843

"Detecting Dotted Lines and Curves in Random-Dot Patterns", Richard Vistnes; Stanford University 849

"Learning Shape Computations", John (Yiannis) Aloimonos and David Shulman; University of Maryland 862

TABLE OF CONTENTS

Page

SECTION III - OTHER TECHNICAL REPORTS (Continued)

"Local Shape from Specularity", Glenn Healey and Thomas O. Binford; Stanford University 874

"Finding Object Boundaries Using Guided Gradient Ascent", Yvan Leclerc and Pascal Fua; SRI International 888

"Detecting Blobs as Textons in Natural Images", Harry Voorhees and Tomaso Poggio; Massachusetts Institute of Technology 892

"Reconstruction of Surfaces from Profiles", Peter Giblin and Richard Weiss; University of Massachusetts at Amherst 900

"Using Occluding Contours for Object Recognition", Willie Lim; Massachusetts Institute of Technology 909

"Parallel Optical Flow Computation", James Little, Heinrich Bulthoff and Tomaso Poggio; Massachusetts Institute of Technology 915

"Using Optimal Algorithms to Test Model Assumptions in Computer Vision", Terrance E. Boult; Columbia University 921

"A Parallel Implementation and Exploration of Witkin's Shape from Texture Method", Lisa Gottesfeld Brown and Hussein A. H. Ibrahim; Columbia University 927

"Localized Intersections Computation for Solid Modelling with Straight Homogeneous Generalized Cylinders", Jean Ponce and David Chelberg; Stanford University 933

"Line-Drawing Interpretation: Straight-Lines and Conic-Sections", Vic Nalwa; Stanford University 942

TABLE OF CONTENTS

Page

SECTION III - OTHER TECHNICAL REPORTS (Continued)

"Line-Drawing Interpretation: Bilateral Symmetry", Vic Nalwa; Stanford University 956

"Estimation and Accurate Localization of Edges", Fatih Ulupinar and Gerard Medioni; University of Southern California 968

Edge-Detector Resolution Improvement by Image-Interpolation", Vishvjit S. Nalwa; Stanford University 981

"Detection, Localization and Estimation of Edges", J.S. Chen and G. Medioni; University of Southern California 988

AUTHOR INDEX

NAME	PAGE
Aggarwal, R.	122
Allen, P. K.	71, 392
Aloimonos, J. Y.	1, 552, 862
Anandan, P.	719
Arkin, R. C.	417
Bajcsy, R.	194
Baker, H. H.	843
Bandyopadhyay, A.	552
Barnard, Stephen T.	792
Beveridge, J. R.	257
Bhanu, B.	122, 466, 581
Binford, T. O.	18, 234, 599, 644, 766, 874
Blelloch, G.	628
Bogdanowicz, J. F.	310
Bolles, R. C.	12, 843
Boult, T. E.	71, 921
Bradstreet, J. D.	127
Brolio, J.	178
Brown, C. M.	65, 663
Brown, L. G.	927
Bulthoff, H.	915
Burger, W.	581
Canning, J.	242
Cass, T.	628

AUTHOR INDEX (Continued)

NAME	PAGE
Chang, S.	507
Chelberg, D.	340, 933
Chelberg, D. M.	447
Chen, J. S.	988
Choi, D. J.	639
Chou, P. B.	663
Chou, T.	704
Cochran, S. D.	777
Cole, W.	272
Collins, R. T.	178
Cooper, P. R.	381
Cyrluk, D.	798
Daily, M. J.	78, 87
Davis, L. S.	1, 153, 507
Dementhon, D.	153
Draper, B. A.	178
Fan, T. J.	351
Feldman, J. A.	65
Fischler, M. A.	12
Freudenstein, D. G.	589
Fua, P.	227, 888
Gajulapalli, R.	153
Giblin, P.	900
Glazer, F.	733

NAME	PAGE
Glicksman, J.	107
Goto, Y.	440
Gremban, K. D.	127
Griffith, J.	178, 521
Hanson, A. J.	227, 475
Hanson, A. R.	55, 178, 417, 538
Harris, J. G.	87
Harvey, W. A.	205
Harwood, D.	507
Healey, G.	599, 874
Hollbach, S. C.	381
Horwedel, M. S.	303
Huertas, A.	272
Huttenlocher, D. P.	370
Ibrahim, H. A. H.	927
Ikeuchi, K.	321
Kamgar-Parsi, B.	671
Kanade, T.	32, 143, 614
Kanatani, K.	704
Kapur, D.	798
Kender, J. R.	71, 399, 574, 589
Kim, J. J.	242
Klinker, G. J.	614

AUTHOR INDEX (Continued)

NAME	PAGE
Kohl, C. A.	538
Kriegman, D. J.	407
Krotkov, E.	194
Kushner, T. R.	153
Lawton, D. T.	107, 447, 497
Leclerc, Y.	888
Lehrer, N. B.	521
Le Moigne, J.	153
Levitan, S. P.	483
Levitt, T. S.	107, 447
Lim, H. S.	234, 644
Lim, W.	909
Little, J. J.	628, 915
Lowry, M. R.	835
Mann, W.	340
Marimont, D. H.	843
McConnell, C. C.	107, 497
McKeown, D. M.	205
Medioni, G.	351, 968, 988
Mintz, M.	194
Moerdler, M. L.	574
Morgenthaler, D. G.	127
Mundy, J. L.	98, 798
Nalwa, V. S.	942, 956, 981

NAME	PAGE
Nasr, H.	432
Nathan, M.	127
Nelson, P. C.	107, 447, 497
Nevatia, R.	8, 272, 351, 360
Nguyen, V.	798
Olin, K. E.	78
Panda, D.	122
Pentland, A. P.	475
Poggio, T.	41, 825, 892, 915
Ponce, J.	340, 933
Price, K. E.	694
Quam, L. H.	475
Raju, G. V. S.	766
Rao, K.	360
Reiser, K.	78, 87
Reynolds, G.	257, 521
Riseman, E. M.	55, 178, 417, 538
Rosenfeld, A.	1, 242, 298, 620
Schaffer, S.	432
Shafer, S. A.	143, 614
Shariat, H.	694
Shekhar, S.	766
Sher, D.	655
Shulman, D.	862

NAME	PAGE
Silberberg, T. M.	313
Smith, E. M.	399
Smith, G. B.	170
Snyder, M. A.	681
Stentz, A.	440
Strat, T. M.	170
Symosek, P.	466
Triendl, E.	407
Tseng, D. Y.	310
Thompson, D. W.	98
Thorpe, C.	143
Ullman, S.	370
Ulupinar, F.	968
Veatch, P.	153
Verri, A.	825
Vilnrotter, F. M.	78
Vistnes, R.	849
Voorhees, H.	892
Weems, C. C.	483
Weiss, I.	552, 749
Weiss, R.	900
Wolff, L. B.	810, 821

SECTION III

OTHER

TECHNICAL REPORTS

Vision and Visual Exploration for the Stanford Mobile Robot

Ernst Triendl and David J. Kriegman

Artificial Intelligence Laboratory
Stanford University
Stanford, CA 94305

ABSTRACT

Soft modeling, stereo vision, motion planning, uncertainty reduction, image processing, and locomotion enable the Mobile Autonomous Robot Stanford to explore a benign indoor environment without human intervention. Details of the first successful run are presented.

1. Introduction

By the end of 1986 the Mobile Autonomous Robot Stanford (MARS) or Mobi was able to move relatively freely under its own guidance inside our laboratory building.

The rational behind the vision and motion planning system is to:

1. be fast enough to allow perceptible movement, and even fast motion when special hardware is added in some distant future
2. to understand enough about a benign indoors environment to move about and recognize the more important elements of a building.
3. to move under its own guidance, exploring rooms without human intervention.

These goals are achieved by a combination of modeling and vision. Modeling tells us that buildings have walls, doors, floors and windows that obey certain relations to each other and the vehicle. The vision system determines the regions of free space and location of obstacles. The motion planning system generates moves that are likely to increase the knowledge about free space so that further moves will be possible.

The examples in this paper are from the first convincingly successful run on Friday, October 18, 1986. Left to its own devices Mobi moved down the hallway, made a tour of the lobby and came back into the hallway, traveling a total distance of 35 meters.

We will now describe the robot, the model and its implication for vision, the vision system, free space and motion planning with uncertainty propagation and finally show details of Mobi's excursion.

2. The Robot

The vision, modeling and navigation system described in this paper was implemented on an omnidirectional vehicle. Mobi [fig 1] is propelled by three wheels that are mounted on the edges of an equilateral triangle. Each wheel has passive rollers along its chord, permitting the wheel to move sideways while being driven. This allows simultaneous rotation and translation over relatively smooth terrain. A motor drives each wheel and a shaft encoder counts the number of rotations. The counts are mapped through kinematics of the vehicle and integrated to determine the change in the robot's position.

This robot has four sensing modalities: vision, acoustics, tactile and odometry. The stereo vision system uses two cameras mounted on a pan/tilt head to learn the details of its environment. The acoustic system is composed of twelve polaroid sensors equally spaced around the circumference of the robot. These sensors return a distance

Support for this work was provided by the Air Force Office of Scientific Research under contract F33615-85-C-5106, Image Understanding contract N00039-84-c-0211 and Autonomous Land Vehicle contract AIDS-1085S-1. David Kriegman was supported by a fellowship from the Fannie and John Hertz Foundation.

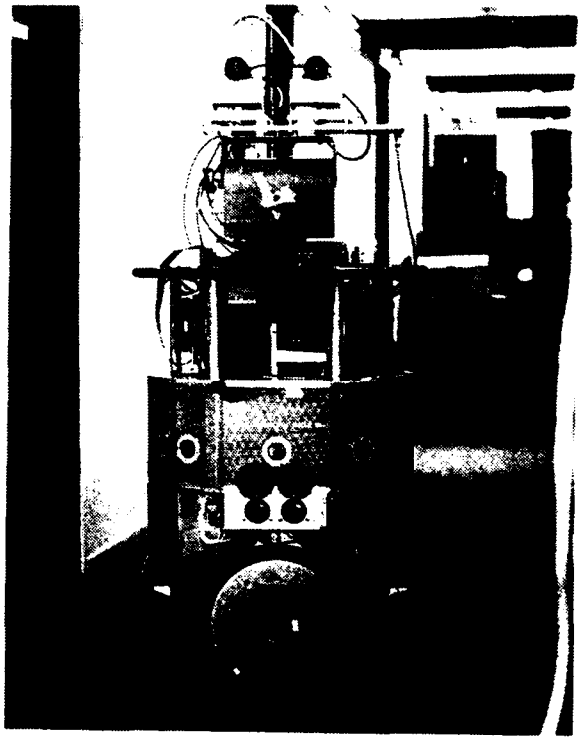


Figure 1. MARS aka Mobi

that is proportional to the time of flight of the echo of an acoustic chirp which the sensor emits. Essentially, the sensor finds the distance to the nearest object within a 30° cone. This information can either be integrated into a map of the environment or simply used for guarded moves against the unexpected. The tactile sensing system in the bumpers uses segmented tape switches to determine the point of contact in a collision. Besides being used for emergency collision detection, tactile sensing has also been used for moving through doorways; the edges of the doors are too close to the robot for the vision and acoustic systems to sense so only odometry and contact sensors are available. More details about how the acoustic and tactile systems are used can be found in [Kriegman].

Mobi's distributed computational system resides partially on the vehicle as well as offboard. Onboard, each wheel is controlled by an 8 bit microprocessor, running a simple position controller. These controllers are commanded by a sixteen bit computer (National Semiconductor 32016) which is also responsible for trajectory planning, sensor data acquisition and offboard communication. Most of the "intelligence" is performed offboard with the onboard computer executing simple commands and handling real time emergency situations such as stopping when the tactile bumpers detect a crash. The robot communicates, via a digital radio link, with a (necessarily offboard) Symbolics Lisp Machine which is responsible for planning and building a world model. A TV transmitter sends the video signal from the cameras to the digitizer. From there the image is transferred, via DMA link, to a VAX which does image processing and computes stereo correspondence points that are sent to the Lisp Machine over an ethernet.

3. Modeling and the Environment

We expect the robot to explore a benign indoor environment without human intervention. A model is used to represent knowledge about this environment and helps to execute tasks within it.

The base model consists of a flat floor that carries vertical, straight walls with hinged doors. Arrangements of walls form rooms and hallways. Walls are very likely to be parallel or at right angles with respect to each other.

In the actual implementation we allow surface marks on the walls and a class "Other Object" to cope with things that the stereo vision system sees, but modeling can not explain, and the motion planner should not push over.

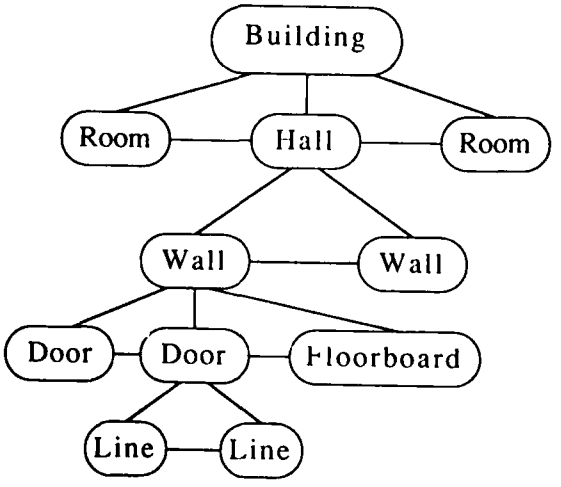


Figure 2. Hierarchy of Model

Figure 2 shows the hierarchy that might describe a hallway in a building. At the lowest level are the points and lines found by the sensors. This information is fit to a model of generic objects such as walls, doors, and the floor. Two parallel walls that bound an elongated region of free space could be a hall, and hallways are found in buildings. So, when searching for a route between rooms in a building, we can first search at the building level and then find a path along successive levels of the map. A graphic of an instantiation of this model containing all possible objects and their joining edges, is drawn in fig 3.

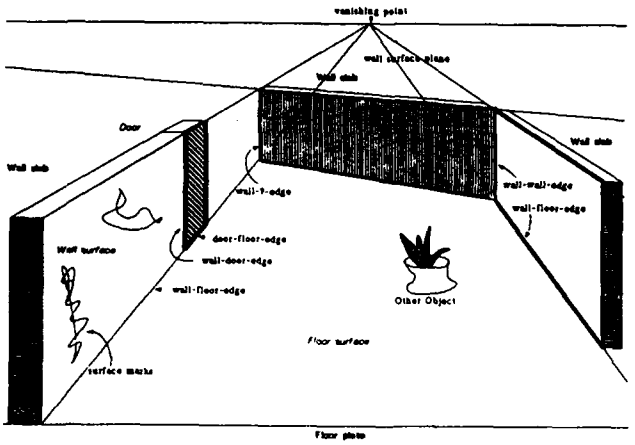


Figure 3. Instantiation of Model

This choice of model has implications for the vision system: All edges of structural elements in our model are either vertical or horizontal. Note the edges in the stereo pair of images in figure 4. All vertical edges cross a horizontal plane at camera height (otherwise Mobi could not fit through doors). Vertical edges alone allow the instantiation of the model with one ambiguity: We may confuse a gap in the wall with the wall, unless some other object is seen through the gap. Looking at the floor edge would possibly resolve this situation.

Whether one should look at the floor at all is also a question of good usage of processing resources: When all information about the model can be gained by looking horizontally, looking down will slow the process (considerably as it turned out). On the other hand looking down occasionally is needed to avoid obstacles on the floor. Our Robot does not do so now and consequently rams into chairs, flower pots (if small) and couches.

4. The Vision System

Image Preprocessing and Edge Detection

Images from both cameras are digitized with a resolution of 256 picture elements per line. A one dimensional version of the edge appearance model is applied to an epipolar line produced by averaging a few lines at the horizon, causing vertical smearing of the images. (The horizon is known from the calibration program.)

This vertical smearing has the following effects:

1. Vertical edges retain their acuity.
2. Slanted edges get blurred, horizontal edges vanish.
3. Tilt and roll angle misalignment have less effect.
4. Image noise is reduced.
5. Blobs become vertical edges.

The last effect can be applied to a different scenario: A busy road at night with the glare of headlights. After passing the vertical averaging filter, these blobs of light would become stripes that cross the horizon and give rise to edges suitable for our algorithm.

Vertical edges of walls and doors in our model remain vertical and sharp after this operation and will be recognized with high localization precision by the edge appearance operator.

The edge appearance model [Triendl 1978] compares a local patch of digital image, e.g. a 5 by 5 pixel area, to the image that would have been created if the camera were looking at an ideal step edge. It uses the spatial filter created by the lens-camera-digitizer-preprocessor pipeline for this purpose. The operator returns quality, position, direction, left and right grey levels of the edge element. It has infinite spatial resolution (1/8 pixel in practice) and degrades gracefully for non-ideal edges.

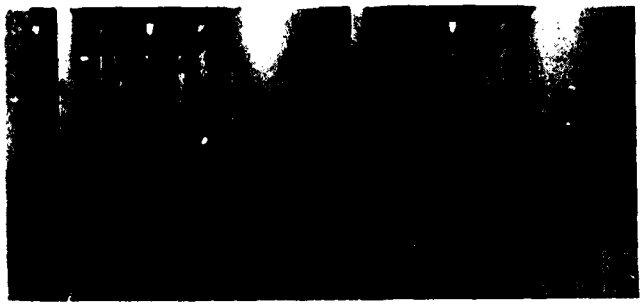


Figure 4. **Stereo pair of images seen by Mobi**

For locations without ideal edges but high gradient the same algorithm is applied at half resolution, because some structural edges may not be sharp enough (rounded corners) or ill defined in detail (reflections or fine structures along a corner). These edges, multiple edges that cannot be combined and locations with a high gradient that to not correspond to ideal edges, have a larger estimated localization error.

The algorithm is trimmed for speed and takes less than one second on a VAX.

Alternative Edge Extractions

Before arriving at the above solution we explored several alternatives. Some of them are still available to be applied later for more specific purposes such as looking for a floor edge

The first experiment was to apply the edge detector to the whole image and group and link edges afterwards into lines. Even after transferring the first stages of the edge detector, several 5 by 5 convolution filters, to a pipeline processor, it took several minutes to process a stereo image.

The next alternative was to combine edge detection and linking. The next edgel is expected at a position predicted by the previous one. Processing time was reduced to 15 seconds per image. Most of the processing went into the search for new edge links. Reducing this search risks loosing short and marginal chains of edges.

The resulting lines were labeled 'straight' and 'bent' and connected with other lines forming corners, T-junction, Y-junctions and arrows. These were intended to label the resulting line-graph according to the model and combine monocular and binocular stereo. Again processing time was too long, and in addition lines were easily broken by door knobs, labels on the wall and the likes, so that stereo pairs were difficult to determine. Some objects, e.g. the previously mentioned flower pot, did not give sufficiently good and consistent edges for a stereo match.

Non-vertical lines that fit the model are floor-wall and floor-door edges. We found that these edges to be often outside the field of view of the camera, too low in contrast, ill defined, or obstructed by furniture. When extracting vertical lines the image can be enhanced by application of a vertical low pass filter, i.e. a moving average over a few lines. The line follower won't get thrown off by a tack in a door frame. All important information, except edge length, is provided by the first edge of a vertical. So finally we drop angular sensitivity of the edge detector and line following and arrive at our present solution.

Figure 5. Stereo pair with occlusions

Stereo Algorithm

We can never be certain that a stereo match is correct, since for any possible stereo match of edges, we can find a possible if far fetched physical explanation. Even in the series of pictures from our test run discussed below, we find matching edges that appear very different, for example in figure 5. What we can do is tap all available sources of information and cues to make the chance of correct matches in the real world as high as possible. See [Baker 1982] and [Tsuji 1986] for other solutions.

The stereo mechanism along an epipolar line that we finally settled on uses edges, grey levels, correlation of intensities, constellations of edges and constraint propagation. It tries to preserve left to right ordering of edge matches but allows violations, e.g. by a pillar in the middle of a room. Multiple choices of matches are kept along if needed.

In the first stage all possible stereo edge matches weighted by

- similarity of edge step size (positive or negative weight)
- grey level similarity on left and right side of the edge (only positive weight). This handles those occluding edges which have similar grey levels on one side only.
- similarity in grey level correlation near the edges. This measure is independent of the widely varying camera gains and offsets and responds to differences in edge cross sections.

Matches with a positive weight are considered candidates at this stage. Note that we did not loose matches of only one side of the edge and that we keep matches with similar edge cross sections even if their grey levels are different. A sample of the resulting situation [figure 7] shows edges and stereo matches considered at this point superimposed to the grey level curve of left and right epipolar line of a sample scene.

The grey level comparison function that we use is similar to the normalized cross correlation but takes differences in standard deviation and mean grey levels into account. Since intervals along the epipolar line may not have the same length and the interval boundaries may be edges at subpixel accuracy, grey levels are resampled by linear interpolation at approximately pixel steps. The ratio of interval lengths also contributes to the comparison function since it is unlikely to see the same surface from very differing angles and similar interval lengths are therefore likely.

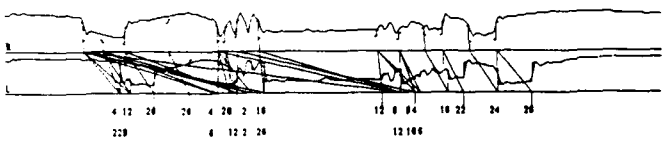


Figure 6. Stereo match proposals and grey level curves

Next we deal with local consistency. First the grey-level-comparison-function is applied to the intervals between pairs of matching edges and their respective first and second neighbors to the left. By comparing these 4 combinations per match, marginal edges present in only one image can be pinpointed and eliminated. As a side effect, high correlation makes it likely that the ribbon between edges results from a solid object.

Local consistency links neighbors and can tell us which neighboring match is more consistent. These neighbor links form paths of maximum consistency that link groups of likely match choices. Simply summing up match qualities in a group represents an effective means of constraint propagation. A group of equal bars or a checkerboard that is entirely visible will be matched correctly this way. In the case of an occluding edge the constraint will propagate up to the edge, and the part visible only to one camera will tend to be left unmatched.

Finally, if a previous image is available, a motion match that occurs within the limits of motion prediction, may resolve any remaining ambiguity.

Matching to the Model: Walls and Doors

Every pair of matched points derived from neighboring edges is potentially part of a wall. Angles of the lines connecting these points are clustered with weights equal to their distances. The floor is then searched for linear constellations of points along angles corresponding to cluster maxima. Straight lines are fitted to these constellations and the best fit is considered to signify a wall surface. Other fits with the same angle or perpendicular to that angle are also considered to be a wall surface. Walls are further labeled 'leftwall' and 'rightwall' if they pass by the left or right side of the vehicle respectively.

Any pair of points 60cm to 140cm apart is a candidate for a door frame. If the corresponding edges have big brightness differences with the dark side and the inner side of the frame a door is assumed. It is confirmed if it is on a wall. If the space between those edges does not have a strong grey level correlation the door is not closed.

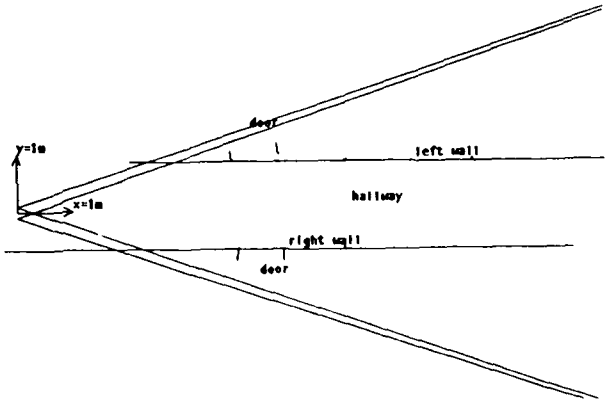


Figure 7. shows the state of the model as derived from a single view of the hallway of our lab. An additional classification has been introduced here: 'hallway' for a room that is very narrow and long.

5. Free Space and Motion Planning

Free space is the volume, or in our case, the floor area, that is free from obstacles. Initially it is the area occupied by the vehicle, a roughly circular polygon 65cm in diameter. The arrangement of robot, cameras and fields of view is drawn in figure 8.

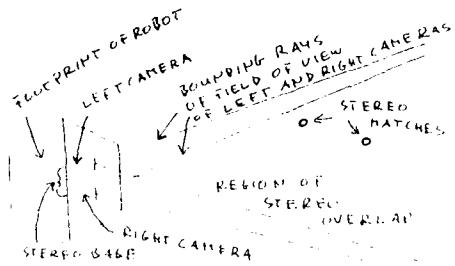


Figure 8. Geometry of robot, cameras and field of view.

When the stereo system finds a match the rays between both cameras and the object go through free space if the match was correct and the physical light rays did not go through glass panes or were reflected by a mirror. The stereo system delivers matches in a polar sort about the robot. We now make the worst case assumption that the connection between neighboring matches represents a solid object. This includes real walls with both ends visible, openings in walls without objects seen through them and lines connecting otherwise unrelated matches.

The hull of lines connecting neighboring matches and lines connecting matches with each camera represents our visual free space [figure 9]. Total free space is the 'OR'ing of the footprint of the vehicle with free space derived from vision and other sources.

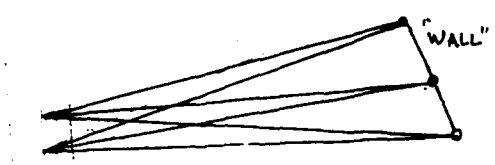


Figure 9. Free space and stereo matches

A safe move will keep the vehicle entirely in free space. From figure 9 we notice that no safe move is possible from a single view since there is a bottleneck between footprint and visual free space.

If the robot has already moved, the new free space consists of the superposition of the new visual free space to the previous visual free space and the space swept out during the motion. Old freespace has to be shrunk according to motion uncertainty.

If the robot has already moved, the new free space consists of the superposition of the new visual free space to the previous visual free space and the space swept out during the motion. Old freespace has to be shrunk according to motion uncertainty.

To allow Mobi to move we tell it initially that it has enough space to rotate in place and position it accordingly at least 5 cm from the next obstacle. The motion planner's top goal is to generate enough free space to allow motion. In particular the vehicle will rotate until it has seen enough to be able to find a save forward. The motion planner works as follows:

1. Intersect free space with a circle with the diameter equal to twice the maximum step size whose center is the robot.
2. for each intersecting arc find the longest legal move in the direction of the center of the arc.
3. the robot's orientation is determined by repeating steps 1 and 2 from the new position. At the first position the robot will face the second one.
4. If there are several moves, or several orientations make a random choice.
5. If no move is found rotate.

Uncertainty Reduction

As the robot moves, its location is determined by odometry. However, due to wheel slippage, finite shaft encoder resolution, backlash, as well as other problems, there is some uncertainty in measuring wheel positions. This is especially significant with odometry because error is integrated with successive motion. Furthermore, there is some uncertainty in measurement of the location of correspondence points from stereo vision. Thus, when the robot views a point in space (e.g. a door frame) from two locations, the point will appear to be at two distinct locations unless one considers uncertainty [Brooks,Chatila]. For some measurements, the easiest solution may be to ignore it by just checking fixed intervals about any point (e.g. two points are the same if they're within 2 inches). A more sophisticated approach than representing points by their coordinates, is to represent them as the probability that the point is at certain location (i.e. probability density function). In this work, all points are represented

by a multivariate normal distribution with a mean vector $\vec{\mu}$ and covariance matrix, Σ. Higher order moments or other distributions may provide more information but are much more complex and would require great computation time.

All stereo correspondences are made at a location P_i which can be related to the robot's previous location P_{i-1} by a transform, $T_{i-1,i}$. Since mobi only travels inside buildings, and the floor is basically level, the transforms only need to represent the Cartesian location and the angle of rotation, (x, y, θ). From odometry, we can determine the mean and covariance of the motion which we have taken to be a function of distance traveled. The stereo measurements are also taken to have uncertainty due to localization of edges in the two images. It should be noted that the standard deviation of the error in distance to a point grows as the square of the distance to the point whereas localization uncertainty in the directions parallel to the cameras' focal plane grows proportional to distance.

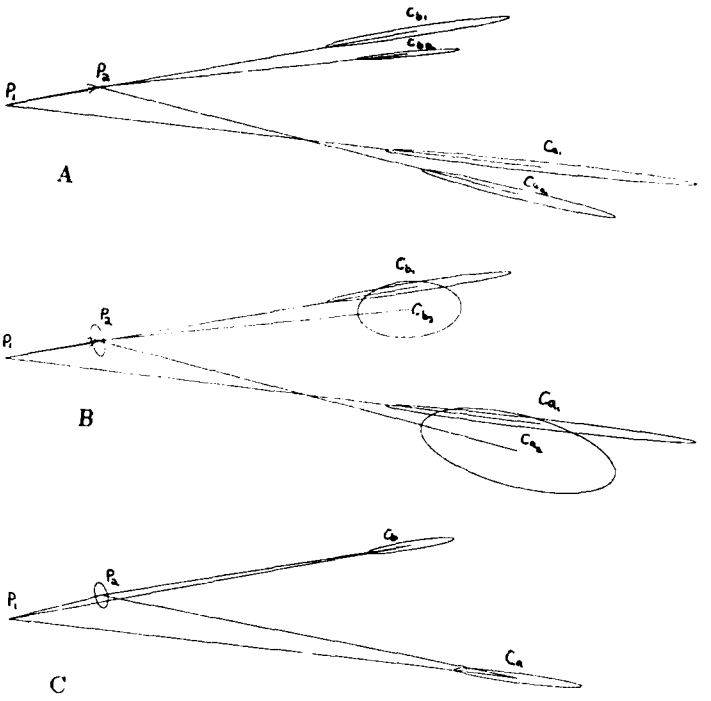


Figure 10. Uncertainty reduction.
A: Two points seen from different locations
B: Uncertainty with respect to P1
C: Uncertainties combined

Figure 10A shows the uncertainty ellipses of two points (C_a and C_b) that were viewed from two different locations, (P_1 and P_2). This figure shows that there was an error in motion transform, $T_{1,2}$ as determined by odometry because correspondence points C_{a_1} and C_{a_2} should be at the same location as well as points C_{b_1} and C_{b_2}. However, consider the uncertainty in the location of the robot at P_2, as represented by the ellipse about its location in fig. 10B. The uncertainty in the location of the stereo points C_{a_2} and C_{b_2} with respect to location P_1 are found by compounding [Smith], also shown in fig. 10B. Now, we can determine that indeed points C_{b_1} and C_{b_2} are actually measurements of the same point, and so we can now apply a Kalman filter to reduce the uncertainty in both the stereo correspondences and the motion transform. The transform and the correspondence points as seen from P_1 become our initial state vector along with their covariances. We assume that the stereo measurements as well as the transform are independent. The correspondence points as seen from P_2 becomes the measurement vector, and we just apply the Kalman filter [Junkins] to obtain figure 10C. Note that the uncertainty of the motion has been reduced as has the location of the correspondence points as viewed from both positions. With this, we can integrate motion sequences to a greater level of accuracy.

Navigation with positions derived from correspondence points proved to be much more precise than odometry.

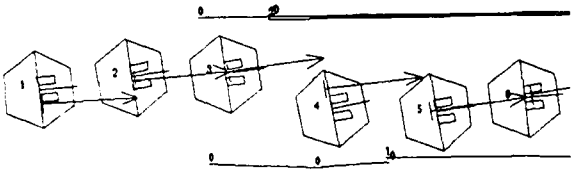


Figure 11. Example for visual navigation: The robot ikons show the derived position, while the arrows indicate predicted motion relative to the previous position. The arrow tails indicate the commanded position of the robot while the direction to the arrow head indicates the commanded orientation. The differences between the arrow tails and the ikon centers are very large and the precision can be judged from the fact that the wall segments as seen from various positions superimpose.

6. The Great Excursion

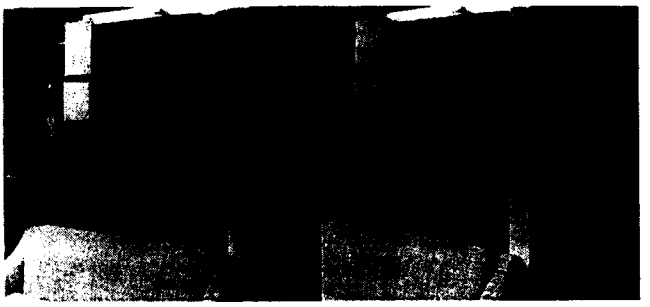
Now, let us take a look at what a real mobile robot run is like. Mobi is turned on in front of the calibration chart discussed previously, and the camera parameters are determined. Mobi is now moved out into the hallway (right now, we don't have the precision to send it through doorways). To make the start of this run interesting, we pointed mobi at a blank wall and commanded it to go; this was our last action, now mobi was on his own.

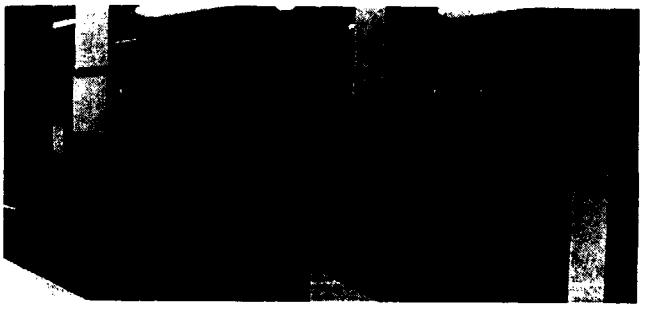
Sure enough, it did not see any edges while staring at a white wall, and so the robot performed the only safe move, rotation. After this move, mobi still hasn't seen enough to move forward and rotates further. It is confronted with its first substantial view, the hallway as shown in figure 4.

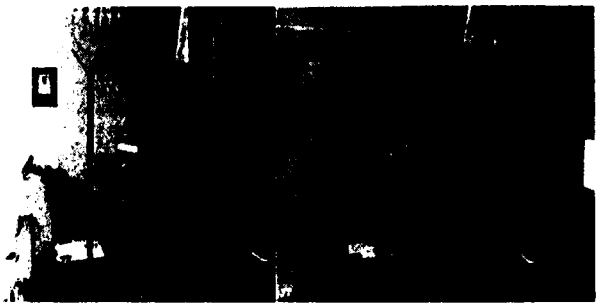
From here it sees enough correspondence points to build a large enough region of free space to begin translating. Mobi moves forward a meter though not yet towards the center of the hall because it needs to see more points on the right wall. After a few wavering moves, mobi travels towards the end of hallway without any further incidents.

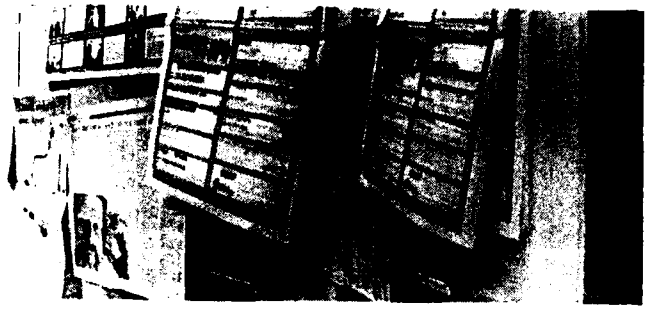
The first great challenge looms off in the distance, the great white pillar (shown in figure 5). This poses a particularly interesting problem because the epipolar ordering constraint may be violated. Furthermore, the building corner causes a major occlusion of the window. Note the white signs on the window are visible in the right image and are occluded in the left image. The occluding edge changes from light gray to black in the left image while switching from gray to bright white in the right image. Also note the black wire dangling close to the corner. This the real world after all. The robot still sees a large region of free space and heads towards the water fountain. Finally, mobi gets close to the end of the hallway, sees little and decides to round the corner as shown in the motion sequence of figure 12 on this page.

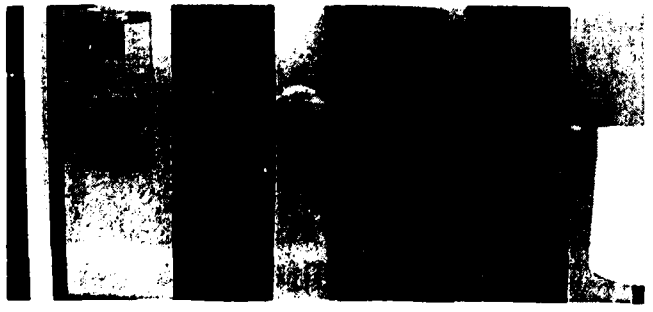
In the first stereo pair, the pillar is seen occluding the paper signs in the right image but not in the left, causing edge inversion again. Still it is recognized, and Mobi passes the pillar. Seeing the reflections of lobby which we'll encounter in the next motion sequence, Mobi heads straight toward the window. Fortunately for us (windows are expensive), mobi sees the edges of the aforementioned paper signs and the two bright dots that are coincidently











at eye level. Mobi rotates again to avoid bumping the window and one of the proud authors is seen in the reflection, smiling from ear to ear. Rotations continue, and we are now facing the far end of the lobby, featuring two Stanford Arms of old, barm and yarm.

Mobi's excursion continues across the lobby heading to window on the other side of lobby. However, as it neared the window, mobi saw its own reflection in the window and turned away. Unfortunately for the authors, this and a few preceding images have been destroyed, but we can still go to the video tape. Now we enter the last image sequence where the window is filling the right image and a bulletin board is dominating the left image. The disparity is nearly half an image because we are so close. Of course mobi rotates and faces a similar scene. This scene is cluttered but fits our model of a wall with surface marks. Rotations continue with the far door coming into view. Mobi is now looking at a large area of free space and moves forward towards that door.

Sadly, the run ended because of a communication failure between the lisp machine and the robot permitting the researchers and onlookers to retire for the evening after celebrating over champagne.

7. Conclusion

So, we have seen that Mobi can successfully navigate through the inside of a building under automatic visual control. The algorithm used during the discussed excursion took between 12 and 18 seconds per vision step plus motion time; Though our choice of algorithm was aimed at producing a lively rate, neither the hardware nor coding was optimized, and too much time was spent on communication between computers.

By choosing a model that can be simply instantiated with the detection of vertical edges in the world, processing time has been greatly reduced. Furthermore, motion sequences have been aggregated to reduce the uncertainty in navigation and stereo. Finally, a symbolic model of building structure has been automatically created.

Aknowledgements

Our thanks go to AFOSR, IU, ALV, FJHF and in particular to Tom Binford, Soon Yao Kong and Ron Fearing for their help throughout this work.

8. References

2. Triendl, Ernst; Kriegman, David, **Stereo Vision and Navigation within a building,** to be published in *IEEE conference on Robotics and Automation*, 1987.

3. D.J. Kriegman; E. Triendl; Tom Binford, **A Mobile Robot: Sensing, Planning and Locomotion**, To be published in *IEEE Int. Conf. Robotics & Automation*, 1987.

4. A.R. de Saint Vincent, **A 3D Perception System for the Mobile Robot Hilare**, *Proc. IEEE Int. Conf. Robotics & Automation*, 1986.

5. S. Tsuji et al., **Stereo Vision for a Mobile Robot: World Constraints for Image Matching and Interpretation**, *Proc. IEEE Int. Conf. Robotics & Automation*, 1986.

6. A.M. Waxman et al., **A Visual Navigation System**, *Proc. IEEE Int. Conf. Robotics & Automation*, 1986.

7. Smith, Randall; Self, Matthew; Cheeseman, Peter, **Estimating Uncertain Spacial Relationships in Robotics**, *Proceeding AAAI Workshop on Uncertainty in Artificial Intelligence*, August, 1986.

8. Brooks, Rodney A., **Visual Map Making for a Mobile Robot,** *Proceedings IEEE International Conference on Robotics and Automation*, 1985.

9. Chatila, Raja; Laumond, Jean-Paul, **Position Referencing and Consistent World Modeling for Mobile Robots** *Proc. IEEE Int. Conf. Robotics & Automation*, 1985.

10. H.P. Moravec, **The Stanford Cart and the CMU Rover**, *Proc. IEEE*, vol. 71, no. 7, July 1983.

11. H. Baker, **Depth from Edge and Intensity Based Stereo** *AIM-347*, Stanford University, 1982.

12. R.A. Brooks, **Symbolic Reasoning among 3-D Models and 2-D Images** *Ph.D. dissertation*, Stanford University, 1981.

13. Junkins, J.L., **An Introduction to Optimal Estimation of Dynamical Systems** Sijthoff & Noordhoff International Publishers B.B., 1978.

14. E. Triendl, **Modellierung von Kanten bei unregelmäßiger Rasterung** in *Bildverarbeitung und Mustererkennung*, E. Triendl (ed), Springer, Berlin, 1978.

15. —, **How to get the Edge into the Map** *Proc. 4th International Conference on Pattern Recognition*, Kyoto, 1978.

AuRA: AN ARCHITECTURE FOR VISION-BASED ROBOT NAVIGATION

Ronald C. Arkin
Edward M. Riseman
Allan R. Hanson

Computer and Information Science Department, University of Massachusetts,
Graduate Research Center, Amherst, Massachusetts, 01003

Abstract

The VISIONS research environment at the University of Massachusetts provides an integrated system for the interpretation of visual data. A mobile robot has been acquired to provide a testbed for segmentation, static interpretation, and motion algorithms developed within this framework. The robot is to be operated in test domains that encompass both interior scenes and outdoor environments. The test vehicle is equipped with a monocular vision system and will communicate with the lab via a UHF transmitter-receiver link.

Several visual strategies are being explored. These include a fast line-finding algorithm for path following, a multiple frame depth-from-motion algorithm for obstacle avoidance, and the relationship of schema-based scene interpretation to mobile robotics, especially regarding vehicle localization and landmark recognition. These algorithms have been tailored to meet the needs of mobile robotics by utilizing top-down knowledge when available to reduce computational demands.

To facilitate the implementation of these algorithms, the UMASS Autonomous Robot Architecture (AuRA) has been developed. Employing both high-level semantic knowledge and control structures consisting of low-level motor schemas, action-oriented perception and schema-based navigation are being investigated. Initial path planning is conducted through a ground-plane meadow-map which contains semantic and visual data relevant to the actual navigational execution of the path. Information from the meadow-map along with the goals developed by the mid-level path planner are then passed to the pilot, which selects appropriate motor schemas for instantiation in the motor schema manager. Pertinent vision algorithms or perceptual schemas are associated with each instantiated motor schema to guide the vehicle along its way.

Results of both actual robot navigation in an outdoor campus environment and appropriate simulations are presented to show the viability of this AI architecture. These examples concentrate in the three areas of vision research mentioned above.

1. Introduction

The VISIONS group at the University of Massachusetts has an extensive and ongoing research project in the interpretation of real-world images [1,2]. More recently, efforts are being made to migrate many of the concepts developed within the VISIONS system to the application of mobile robot navigation. A mobile robot has been acquired to provide an experimental testbed for this effort. This vehicle (a Denning Mobile Robot - DRV) is equipped with a monocular video camera and a ring of 24 ultrasonic sensors.

This research was supported in part by the General Dynamics Corporation under grant DEY-601550 and the U.S. Army under ETL grant DACA76-85-C-008.

AuRA (Autonomous Robot Architecture) is a system architecture that provides necessary extensions to the VISIONS system that are primarily concerned with safe mobile robot navigation. These extensions include the addition of representations specific to navigation, the incorporation of motor schemas as a means of associating perceptual techniques with motor behaviors, and the introduction of homeostatic control utilizing internal sensing as a means for dynamically altering planning and motor behaviors.

The remainder of this paper is divided into the following sections. Section 2 will present an overview of the AuRA architecture. Section 3 will describe how navigation is accomplished within AuRA, specifically the roles of long-term and short-term memory and the operation of the navigator, pilot and motor-schema manager. Section 4 describes the relationship of VISIONS to AuRA, concentrating particularly on the VISIONS schema system. Modular vision algorithms, including techniques already embedded as well as those currently being investigated for potential incorporation, are described in section 5. Experimental results, including robot experiments as well as simulations, are presented. A summary of the paper and a brief description of future work complete the report.

2. Architecture Overview

A block diagram of AuRA is presented in Figure 1. AuRA consists of five major components: the planning, cartographic, perception, motor and homeostatic subsystems. The planner consists of the motor schema manager, pilot, navigator and mission planner and is described in section 3. A cartographer, whose task is to maintain the information stored in long- and short-term memory and supply it on demand to planning and sensory modules, provides the additional functionality needed for navigational purposes. Long-term memory (LTM) stores the *a priori* knowledge available to the system, while short-term memory (STM) contains the acquired perceptual model of the world overlaid on an LTM context. The cartographer is also responsible for maintaining the uncertainty in the vehicle's position.

A perception subsystem, (ultimately consisting of the VISIONS system, sensor processing and sensors), is delegated the task of fielding all sensory information from the environment, performing preliminary filtering on that data for noise removal and feature enhancement, then extracting perceptual events and structuring the information in a coherent and consistent manner, and finally delivering it to the cartographer and motor schema manager. It is also the subsystem, in conjunction with the cartographer, where expectations are maintained to guide sensory processing.

The motor subsystem is the means by which the vehicle interacts with its environment in response to sensory stimuli and

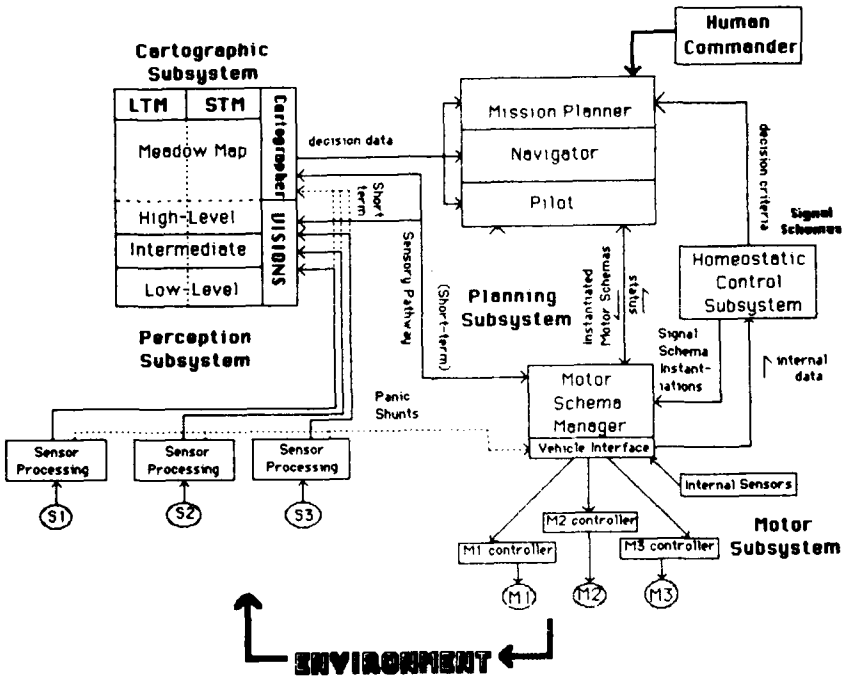


Figure 1. System Architecture for AuRA

high-level plans. Motors and motor controllers serve to effect the necessary positional changes. A vehicle interface directs the motor controllers to perform the requested motor response received from higher level processing.

The homeostatic control subsystem is concerned with the maintenance of a safe internal environment for the robot. Internal sensors provide information which can dynamically affect the decision-making processes within the planner, as well as modify specific motor control parameters.

The first pass implementation of the perceptual system does not draw on the VISIONS system in its entirety. The VISIONS system is ultimately expected to be the location where all sensor fusion occurs, yielding ideally a rich 3-D model of the perceived world. At this stage in AuRA's development, however, relevant vision algorithms are extracted from the VISIONS environment and are used outside of its context. The real-time needs of mobile robotics can be handled by this strategy as the vision algorithms are not yet developed on parallel hardware. In so doing, the cartographer assumes greater responsibility than might be needed in future designs when many of the cartographer's responsibilities are subsumed by the VISIONS system. Figure 2 shows AuRA's initial implementation strategy. Homeostatic control is to be implemented only after the motor schema manager is moved from simulation to real-time implementation and the vehicle is equipped with the necessary internal sensors. The mission planner is currently rudimentary and has a low priority for development.

The subsections that follow describe briefly the ultimate roles of the various AuRA subsystems (with the exception of the planning subsystem which is discussed in section 3.)

2.1 Cartographer

The cartographer is the manager of the non-VISIONS representations and high-level controller of the map maintenance processes. Its responsibilities include:

- Preservation of the integrity of the perceived world model, reconciling temporally conflicting sensor data.
- Initiating and scheduling processes whose duty it is to:

  incorporate data from the perception subsystem into short-term memory(STM)

  instantiate models from LTM into STM

  provide sensor expectations and to guide schema instantiations

- Maintenance of uncertainty at all levels of representation

  spatial error map maintenance for robot localization

  STM environmental uncertainty handling (object location)

- Initial LTM Map building (i.e. knowledge acquisition)

2.2 Perception Subsystem

Environmental sensor processing occurs within the confines of the perception subsystem. This component of AuRA consists of three submodule types: sensors, sensor processors, and the VISIONS system. In the early stages of the robot system development, we utilize a small subset of available vision algorithms, including simplified versions of some low-level algorithms that are tuned for real-time performance, until a full real-time scene interpretation VISIONS environment becomes available.

Sensor processors serve to preprocess the sensor data into a form that is acceptable to the receiving modules. The principal goal for these sensor-specific filters (e.g. from vision, ultrasonic or dead-reckoning sensors) is to simplify the job facing the VISIONS system by removing noisy, extraneous, or errorful data and by converting the relevant data from diverse sensors into a form that is integrable into world representations.

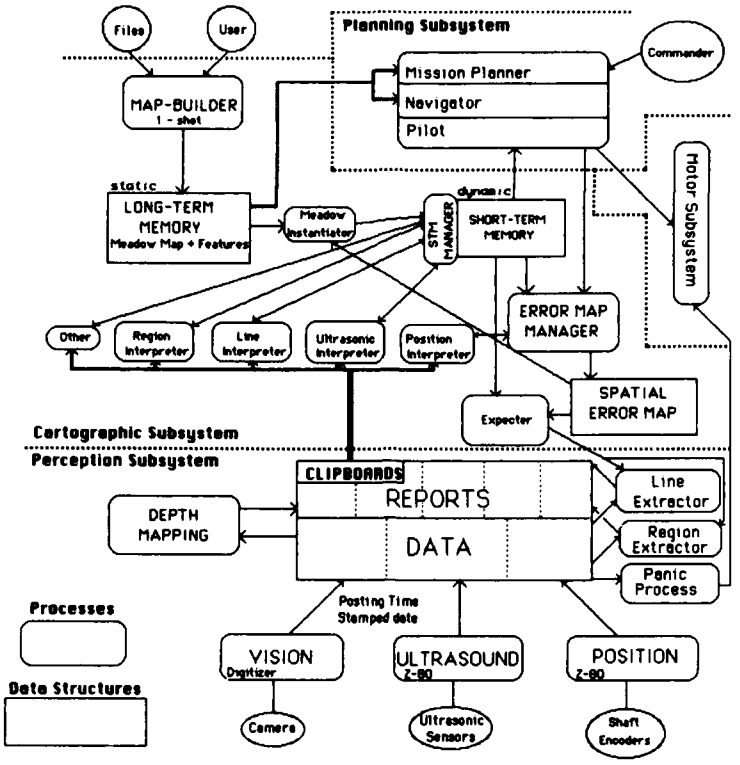


Figure 2. First Pass Implementation of AuRA Architecture

The VISIONS system is the heart of the perception subsystem. Multiple levels of sensor data and their associated interpretations are present. Perceptual schemas are instantiated and maintained within this system. The net result is a collection of plausible hypotheses and interpretations for sensor data with associated confidence levels that reflect their uncertainty. Data can be drawn off by the planner at any representation level within the VISIONS system, ranging from low-level pixel data and intermediate-level lines and surfaces, to high-level full scene interpretations.

Information foretelling imminent danger will pass directly to the pilot or vehicle interface from the sensor processors via panic shunts without the mediation of the cartographer, VISIONS system or motor schema manager. These panic shunts are intended to emulate reflex arc activity, bypassing higher level processing.

2.3 Motor Subsystem

The motor subsystem is delegated the responsibility of effecting the commands of the motor schema manager. The motor subsystem consists of three major components: motors, motor controllers, and vehicle interface. The steering motors, drive motors and motor controllers, in the case of the UMASS DRV, are provided by the manufacturer.

The vehicle interface, in its most general version, will use the output of motor schemas as a means for effecting action in the environment. This module is the one component of the overall architecture which is most profoundly influenced by the specific robot vehicle chosen. Vehicle independence is a design goal for all other AuRA modules. The vehicle interface translates the commands from the motor schema manager into the specific form required for the vehicle.

2.4 Homeostatic Control Subsystem

Concern for behavioral changes in planning due to the internal state of the robot has not been encountered elsewhere in the literature. Most systems assume optimal conditions at all times, others (e.g. [3]) operating in hazardous environments simply determine if it is safe or not to enter a particular location, still others (e.g. [32]) make plans based on fuel reserves and other factors, but the robot's dynamic behavior is not considered.

In the proposed system, internal surveillance of the robot is constantly maintained by appropriate sensors. "Life"-threatening conditions such as excessive temperatures, corrosive atmospheres, or low energy levels, can dynamically alter variables in the motor subsystem and affect decision-making within the planning subsystem. A detailed description of the issues for such a homeostatic control system can be found in [5]. This extended functionality will provide the robot with enhanced survivability through a greater capability to respond to a changing environment.

Although initial system designs will assume optimal conditions for the homeostatic control system, in order to simplify the integration of this concept into later versions, its design considerations will be dealt with from the start.

3. Navigation

Several papers document the navigational and path planning techniques used in AuRA. The roles of the navigator and long-term memory are described in [6] and the motor schema manager's function is presented in [7]. A more complete description of the planning subsystem and navigational representations appears in [8]. The intent of this section is to provide an overview

of the process of navigation for our system, concentrating particularly on the relationship of visual perception to the robot's path choice and successful path completion.

There are two distinct levels of path planning available: map-navigation, based on *a priori* knowledge available from the cartographer and embedded in long-term memory; and sensor-data-driven piloting conducted by the motor-schema manager upon the receipt of instructions from the pilot. The motor schema manager is perhaps best viewed as the execution arm of the pilot, responding to the perceived world in an intelligent manner while striving to satisfy the navigator's goals. First, let's examine the hierarchical planning component of the planning subsystem.

A hierarchical planner, consisting of a mission planner, navigator and pilot (fig. 3), implement the requested mission from the human commander. The functions of the three hierarchical submodules are described below. It should be remembered that communication is two way across the submodule interfaces, but is predictable and predetermined, the central characteristic of hierarchical control.

3.1 Mission Planner

The mission planner is given the responsibility for high-level planning. This includes spatial reasoning capabilities, determination of navigation and pilot parameters and modes of operation, and selection of optimality criteria. Input to this module is from three sources: the cartographer, the homeostatic control subsystem and the human commander. The cartographer provides current world status, including both short-term and long-term memory structures. The homeostatic control system provides data regarding the robot's current internal status: energy and temperature levels and other relevant safety considerations that have a bearing on the robot's ability to successfully complete any plan. No assumptions should be made by the planner that the robot has the necessary resources available to complete any plan that is developed. This is crucial for reliable long-range planning capabilities.

Mission commands are entered by the human commander through a user interface. The exact structure of these commands will be dictated by the task domain (domestic, military, industrial, etc.).

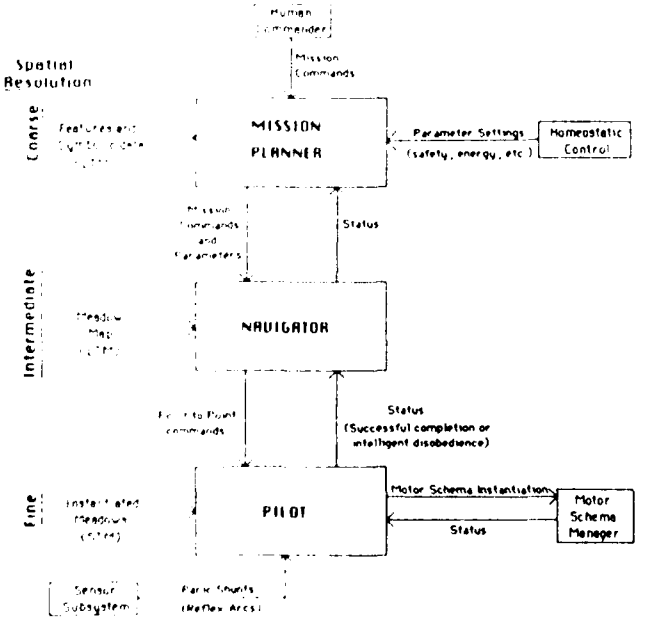


Figure 3. Hierarchical Planner for AuRA

Real-time operation is not as crucial for mission planning as it is for lower levels in the planning hierarchy. Nonetheless, efficient replanning may be necessary at this level upon receipt of status reports from the navigator indicating failure of the attainment of any subgoal.

The output of the mission planner is directed to the navigator. It will consist of a list of parameters and modes of operation that determine the overall behavior of the robot. Additionally, a list of mission specifications and commands (subgoals) for the current task will be provided.

The mission planner, although a significant component of the overall architecture, has a relatively low priority for implementation at this time.

3.2 Navigator

The navigator accepts the specifications and behavioral parameter lists from the mission planner and designs a point to point path from start to goal based on the current *a priori* world model stored in LTM. The representation level used by the navigator is the "meadow map" - a hybrid vertex graph free-space world model. Status reports are issued back to the mission planner either upon successful completion of the mission specifications (subject to the behavioral constraints) or upon failure to meet the requisite goals. If failure results, the reason for failure is reported as well.

The meadow map's basic structure is an outgrowth of work by Crowley [9] and Gmalt and Chatila [6,11,12]. Our work is distinguished from that which preceded it by the incorporation of extensions to include multiple terrain types, the use of specialized map production algorithms, the availability of several search strategies and the ability to easily embed perceptual knowledge. Data stored at this level reflects geometrically and topologically the robot's modeled world. A polygonal approximation of all obstacles is used to simplify both map building and navigational computation. The necessary visual representations (feature map) for path execution and uncertainty management are tied to these polygonal ground plane projection models. This map serves as the basis for the robot's short-term memory context. Specific components are instantiated in STM based upon the robot's current position and the current navigational subgoal.

The 2-D feature map can be viewed as a facet or appendage of the associated meadow map. Data pertaining to the distinctive features of terrain, obstacles, landmarks and other signifiers that are embedded within the meadow map constitute the 2-D feature map. The information stored here contains the attributes of the meadow map's vertices (1-D representations), lines (2-D image representations), and polygons and their associated obstacles or free space (3-D models).

Depending upon the robot's current position, meadows from long-term memory are moved into short-term memory. These contain information on landmarks currently visible, features of known obstacles, terrain characteristics, and the like. This data is available for prediction by the perception subsystem or for use by the pilot for schema instantiation. All meadows visible from the robot's current position and orientation as well as those expected to be visible during the path traversal will be made current in STM.

Output of the navigator is directed to the pilot. This output consists of a point-to-point path and necessary parameters and modes that will affect the pilot's overall behavior. Essentially, the navigator is model-driven, (the model being the meadow map), passing off its goals to the data (sensor)-driven pilot. Status information is received by the navigator from the pilot indicating either the successful completion or failure of the estab-

lished goals of the pilot. Upon pilot failure, the navigator may initiate replanning without reinvoking the mission planner. Time constraints are more critical for the navigator than the mission planner but are not as stringent as those needed for the real-time requirements of the pilot and motor schema manager. See [6] for a complete description of the navigator and meadow-map representation.

3.3 Pilot

The pilot's function is to accept a point-to-point path specified by the navigator and provide the robot with suitable motor behaviors that will lead to its successful traversal. The pilot accomplishes this by selecting appropriate motor schemas from a repertoire of available behaviors (based on the current long-term memory context) and passing them (properly parameterized) to the motor schema manager for instantiation. From that point on, path execution is turned over to the motor schema manager. During actual path traversal, the cartographer concurrently builds up a short-term memory representation of the world based on available sensor data. If, for some reason, the motor schema manager fails to meet its goal within a prescribed amount of time, the pilot is reinvoked and an alternate path is computed by the pilot, based on both the LTM context and STM. Approximating polygons representing sensed but unmodeled (i.e. unexpected) objects are inserted into the local ground plane instantiated meadows and the convex-decomposition algorithms (used by the cartographer to build LTM) are run upon them. These "fractured" meadows serve for short-term path reorientation by the pilot and the basis for the instantiation of new motor schemas.

Associated parameters for the slot-filling of motor schemas will be provided by the mission planner, navigator and LTM. The new commands issued by the pilot will result in motor schema instantiation within the motor schema manager.

Typical motor schemas include:

- **Move-ahead**: Move in a specified direction.
- **Move-to-goal**: Move to an identifiable world feature.
- **Avoid-static-obstacle**: Avoid collision with unmodeled stationary obstacles.
- **Stop-when**: Stop when a specified sensory event occurs.
- **Stay-on-path**: Remain on an identifiable path (road, sidewalk, etc.)

Associated perceptual schemas (run in the context of the motor schema manager) include:

- **Find-obstacle**: identify potential obstacles using a particular sensor strategy.
- **Find-landmark**: Detect a specified landmark using sensory data (for managing the robot's positional uncertainty).
- **Find-path**: Locate the position of a path on which the robot is currently situated using a specified sensor strategy.

3.4 Motor Schema Manager

Distributed control for the actual execution of path travel occurs within the confines of the motor schema manager. Multiple concurrent schemas are active during the robot's path traversal in a coordinated effort to achieve successful path transition. A potential field methodology [13,14] is used to provide the steering and velocity commands to the robot. An overall velocity vector is produced from the individual vector contributions of each active motor schema. This vector determines the desired velocity of the robot relative to its environment. When each motor schema is instantiated, at least one relevant visual algorithm or perceptual schema is associated with it. Figure 5 shows a simulation of a field produced by the instantiation of several independent motor schemas. Additionally, various perceptual schemas are instantiated to identify available landmarks (as predicted by long-term memory and the current uncertainty in the robot's position). These are used to localize the vehicle without necessarily evoking motor action. The role of the motor schema manager, the potential fields representations it uses, and the underlying motivation for its use are presented in [7].

3.5 Navigation Scenario

Perhaps the best way to convey the navigational process within AuRA is by example. Fig. 4a represents an LTM meadow-map model of the area outside the Graduate Research Center at UMASS. Embedded within this map, (although not visible in the figure), is additional data regarding landmarks, building

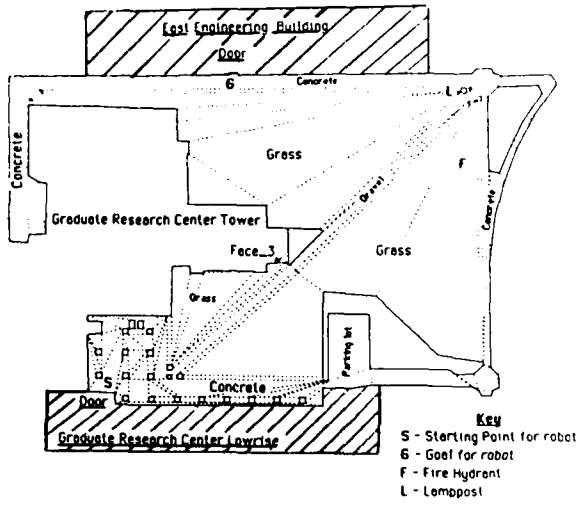


Figure 4a. Outdoor Meadow Map

This map represents the area outside the Graduate Research Center when viewed from above. The detail level of this particular map stored in LTM is low so that small objects are treated as unmodeled obstacles for global path planning purposes.

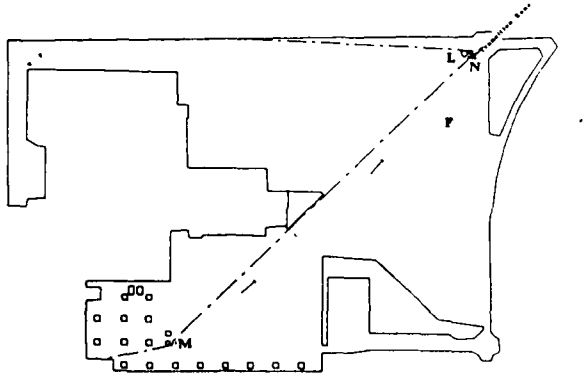


Figure 4b. Global path constructed by navigator

An A* search algorithm is used to search the midpoints and edges of the bordering passable meadows to arrive at the global path.

surfaces, terrain characteristics, etc. This includes specific visual cues to assist the robot during its path traversal.

Suppose the robot is given the command to go from its current position (outside the GRC low-rise) to meet Professor X. Available weather data indicates that the grassy regions are currently impassable (the ground is muddy due to rain), and the robot must restrict its travel to the concrete sidewalks or the gravel path. The mission planner, recognizing this, sets the traversability factors for the grassy regions to IMPASSABLE, locates the fact that Prof. X's office is in the East Engineering (EE) building, determines that he is likely to be in his office at this time (by referring to the current time of day and the day of week) and then invokes the navigator to determine a path from the robot's current position to the door of the EE building. We'll ignore the indoor navigation issues for the purposes of this paper.

The navigator, based on the instructions from the mission planner, determines a global path that satisfies these goals using an A* search algorithm through the meadow boundaries (fig. 4b - see [6] for the details of how this is accomplished). This path consists of 5 legs, the individual piecewise linear components of the path. Let's look particularly at leg 3, where the robot is to follow the gravel path (i.e. assume the robot has successfully traversed the first 2 legs of this path). The pilot receives the message to travel from point M, representing the center of probability of the robot's current position, to N, the end of the gravel path.

The pilot now has available in short-term memory "instantiated meadows" (i.e. those LTM meadows over which the robot is expected to pass during this particular leg of the journey, and several additional visible adjacent meadows, all provided by the cartographer). From this LTM data, the pilot extracts the following relevant facts:

1. Path - The robot is to travel over a gravel path bordered on either side by grass.
2. Landmark - At the end of the path, near where the robot is to turn, is a lamppost.
3. Landmark - Off to the right of the path appears a bright red fire hydrant (a readily discernible landmark).
4. Landmark - To the left of the path, the robot will pass the GRC tower, a 16 story building (another good landmark).
5. Obstacles - It is possible (as always) that people, cars or unmodeled obstacles may be present on the path (either stationary or moving).
6. Goal - At the end of this path there is a change in terrain type, from gravel to concrete.

1 is useful for a path following strategy, **2** and **6** are useful for goal recognition, **1,2,3,4,6** are useful for localization purposes, and **5** is necessary for obstacle avoidance.

From this information, the pilot determines that appropriate behaviors for this particular leg (travel across the gravel path) include:

A. **Stay-on-path(find-path**(gravel))
B. **Move-ahead** (NNE -- 30 degrees)
C. **Move-to-goal**(right(**find landmark**(LAMPPOST_107),3))
D. **Move-to-goal**(**find-transition-zone**(gravel,concrete))
E. **Find-landmark**(HYDRANT 2)
F. **Find-landmark**(GRC TOWER(face.3))
G. **Avoid-obstacles**

Motion is first initiated by the **move-ahead** schema, directing the robot to move in a particular direction in global coordinates, in response to the pilot's need to satisfy the navigator's **subgoal** to move to point N. This heading is based on information contained within the spatial uncertainty map that reflects the uncertainty in the vehicle's position and orientation relative to the world map as well as the specific direction of this particular path leg. It is not critical that the heading be exactly correct; indeed significant error can be tolerated due to the presence of the **stay-on-path** motor schema. As soon as a **move-to-goal** schema becomes active (due to the recognition of the goal – the lamppost and/or terrain type transition zone), the **move-ahead** schema is deinstantiated in favor of it. Motor actions produced by the **move-ahead** schema and **move-to-goal** schema are mutually exclusive.

Stay-on-path(**find-path**(gravel)) yields 2 perceptual subschemas for one motor schema: **find-path-border** - using a line-finding algorithm to detect the position of the path's edges, and **segment-path**, a perceptual schema that uses region-based segmentation to locate the spatial extent of the path. Through

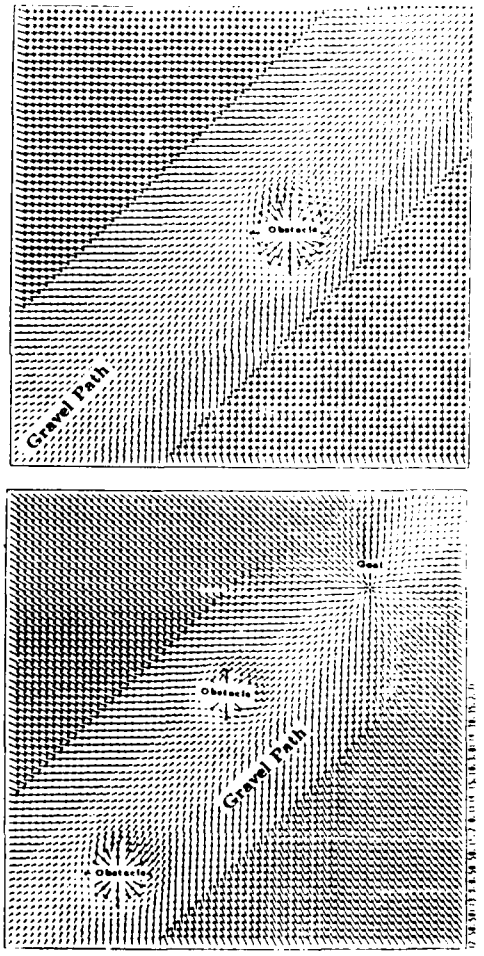


Figure 5. Potential fields produced during leg traversal
The arrows represent the desired velocity vectors that constrain the robot's motion.
a) Before the goal is identified, the **move-ahead** and **stay-on-path** SIs conduct the robot on its way. A single **obstacle** SI is present.
b) After the goal is identified, the **move-to-goal** SI replaces the **move-ahead** SI. Two **obstacle** SIs are shown as the goal is approached.

the combined efforts of these cooperating schemas the position of the path relative to the robot is ascertained. As a result of the posted path position, the **stay-on-path** motor schema produces an appropriate velocity vector moving the vehicle towards the center of the path (fig. 5a).

We will define a schema instantiation (SI) to be the activity of applying a general class of schemas to a specific case [7,33]. The lamppost at the end of the path results in the creation of a **find-landmark** schema(s) dedicated to finding LAMPPOST_107 whose model is extracted from LTM via the instantiated meadows in STM. This **find-landmark** schema directs the sensor processing by instantiating a VISIONS perceptual schema and/or looking for particular strong vertical lines in a given portion of the image and/or utilizing any other relevant sensor algorithm. Every time a potential LAMPPOST_107 is found in the image, (perhaps evidenced by a pair of strong parallel long vertical lines in an appropriate window of the image), a new LAMPPOST_107 SI is created and monitored independently of all other similarly created LAMPPOST_107 schema instantiations. When sufficient supportive data is available confirming that one of the SIs is highly probable to be the landmark desired, all other LAMPPOST_107 SIs are deinstantiated (or placed into hibernation) and the appropriate motor schema (**move-to-goal**) starts producing a velocity vector directing the robot to a point 3 feet to the right of the identified lamppost. If the certainty in the current LAMPPOST_107 drops below a certain threshold, other SIs may be activated or created in response to particular visual events that correlate to the lamppost's model. Additionally, output from the **find-landmark** schema is used to update the robot's spatial error map, independent of any motor action that may result from the **move-to-goal** SI.

The **move-to-goal**(**find-transition-zone**(gravel,concrete)) SI is handled in a similar manner, but different perceptual schemas are instantiated and the image is searched in different regions. Texture measures for gravel are of value as well as the presence of a strong horizontal line within the boundaries of the path. The **move-to-goal** schema contains an implicit **stop-when** schema, so when the target is reached the pilot is notified that the goal has been achieved and the next leg can be undertaken.

The **find-landmark**(HYDRANT_2) schema might involve a color-based segmentation, tagging all bright red blobs in a particular portion of the image as a potential fire-hydrant. Ultimately size and shape from a model of the hydrant would be brought into focus to confirm the hypothesis to prevent incorrect identifications (e.g. a red car, or a person with a red coat). Once identified, this hydrant is then used to reduce the uncertainty in the robot's position (i.e. localization). The same kind of operation would be involved in **find-landmark**(GRC_TOWER(face_3)) but instead of using color as the primary agent for hypothesis formation, a strong vertical line (the building is 16 stories high!) or a corner silhouetted against the sky would be more suitable as the main strategy.

The **avoid-obstacles** schema is actually active most of the time. Simply put, the image is windowed in the direction of the robot's motion and if any unusual events occur in that area (e.g. change in texture, color, strong line, etc.) an **obstacle** SI is associated with that particular event. That portion of the image is monitored over time by the **obstacle** perceptual SI to try to confirm or disprove the hypothesis that the visual event is truly an obstacle. Concurrent with the instantiation of the **obstacle** perceptual schema is the instantiation of an **avoid-obstacle** motor schema. If the monitored obstacle's certainty becomes sufficiently high and the robot enters within the sphere of influence of the obstacle, then a repulsive velocity field is produced by the **avoid-obstacle** SI, altering the robot's course. If, on the other hand, the hypothesized obstacle eventually is determined to be a phantom and not a real obstacle at all, both the perceptual and motor schemas are deinstantiated. When an active obstacle passes outside of the influence of the vehicle, its SIs are deinstantiated as well. Nonetheless, information about the obstacle's position is maintained in STM by the cartographer at least for the duration of the leg traversal.

Figure 5 shows a potential field simulation representative of the robot traversing an obstacle studded path as above. More details regarding the interaction and operation of the motor schemas in AuRA can be found in [7].

4. VISIONS and AuRA

Scene interpretation has long been a primary research effort within the VISIONS group at the University of Massachusetts. A considerable literature exists describing the progress to date [1,2,23,24,29,31]. The remainder of this section will first describe briefly the operation of the schema system, followed by the role that the schema system can play in mobile robot navigation. It should be understood from the onset that schema-based scene interpretation is currently a very time-consuming and computationally expensive process. Work is underway, however, to provide parallel hardware (the UMASS Image Understanding Architecture [29,30]) to speed up this process by several orders of magnitude. Additionally, available *a priori* knowledge present in LTM can be used to guide schema instantiation and reduce the processing requirements dramatically.

4.1 The Schema System

The VISIONS schema system accepts an image as input and produces a labeled interpretation of the observed environmental objects (fig. 6) and, to the degree possible, a 3-D representation of the environment. There are 3 levels of processing available utilizing both bottom-up and top-down processing (fig. 7). Taking a bottom-up view first, the low-level processes operate on pixel level data producing an intermediate symbolic representation of lines, surface and volume tokens. At the highest level, schema processes exist which interpret and collect the intermediate representations into labeled objects.

If no top-down guidance was available, it would be virtually impossible for the system to converge on an acceptable interpretation. Perceptual schemas (in the context of VISIONS) post hypotheses about what specific image events mean. Each highly rated hypothesis guides intermediate and low-level processes in an effort to find self-supporting evidence. This top-down guidance brings the intermediate and low-level processing requirements down to tolerable levels. If the hypothesis cannot find sufficient support or is contradicted by other data, it is deinstantiated. On the other hand, if sufficient support for a hypothesis is available, that particular portion of the image will be labeled as being associated with a particular environmental object and inference mechanisms can direct further semantic processing.

It is quite difficult to describe the operation of the schema system in a few paragraphs. It is hoped that the interested reader will refer to the more comprehensive descriptions cited above [esp. 29,31] for a better understanding of its operation.

4.2 Utilization of VISIONS Schemas in Mobile Robotics

The principal test domains to date for VISIONS schema-based scene interpretation have been house scenes and road scenes (fig. 6). These efforts have been predominantly concerned with full scene labelings with few specific expectations established for the particular image or environment in question other than it being a house or road scene (i.e. there is no world map of the domain).

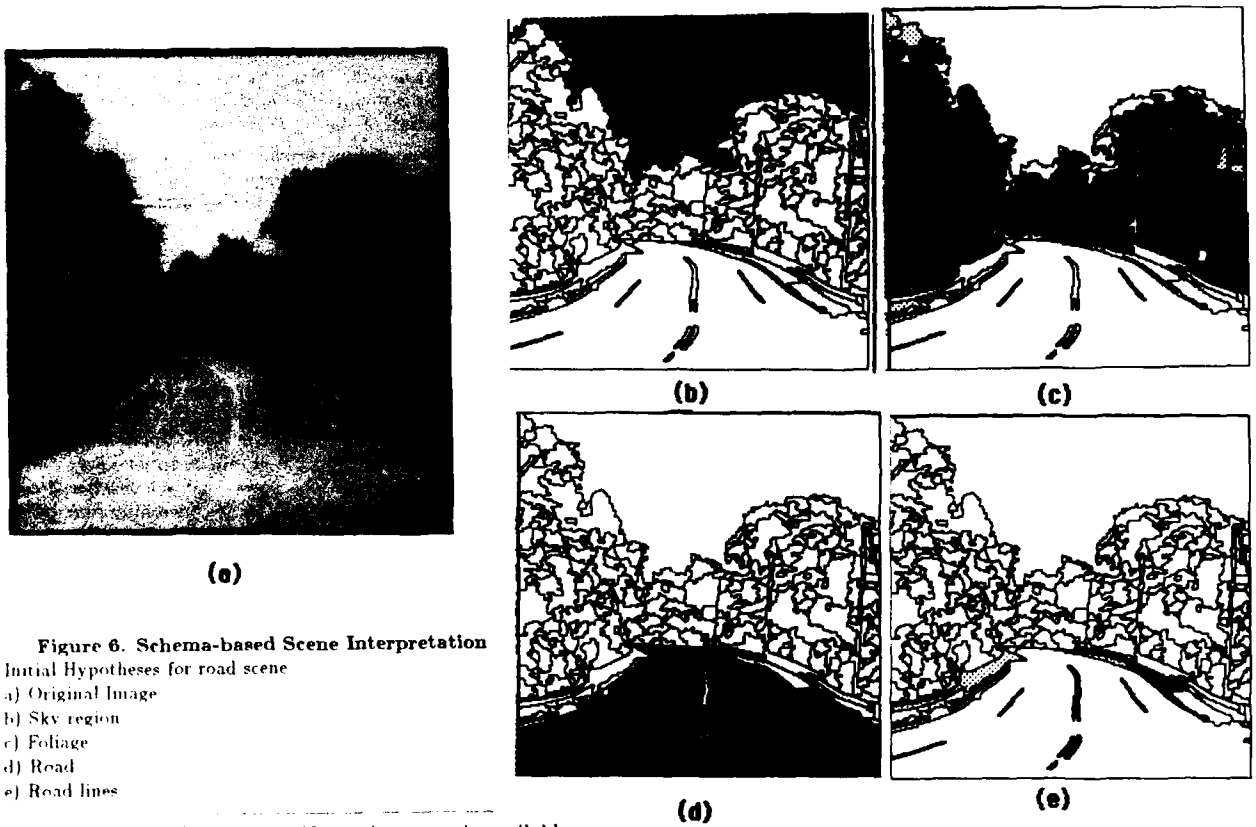


Figure 6. Schema-based Scene Interpretation
Initial Hypotheses for road scene
a) Original Image
b) Sky region
c) Foliage
d) Road
e) Road lines

If *a priori* knowledge of a specific environment is available, it can guide the posting of schema hypotheses, reducing the amount of computing time required to achieve a satisfactory labeling. If the robot's position is approximately known within a global map, this information regarding the potential position of environmental objects (e.g. landmarks or roads) can be used to restrict the formation of object hypotheses to particular portions of the image. The occurrence of two known objects in predicted positions relative to each other can significantly increase the plausibility of a proposed interpretation.

Where can schema-based scene interpretation be used in mobile robotics? In the most grandiose sense, one can say for everything. If a completely and correctly labeled image is available, it can be used for navigation, obstacle avoidance, localization, goal recognition, etc. Indeed the other algorithms described in this paper (line finding, region extraction, etc.) actually constitute some of the lower level processes used within the VISIONS system. Being realistic however, one must recognize that real-time responses are necessary for mobile robot navigation, indicating that schema-based scene interpretation is presently too slow to be effective. A more appropriate current use of the VISIONS schema system would be to provide for the top-down extraction of semantic objects of interest required for several of the other visual processes. If the initial image is analyzed by the scene interpretation mechanisms, it could yield the road edges that can be used to bootstrap the **stay-on-path** motor schema and **find-path** perceptual schema. Additionally it could provide the initial region statistics to seed the region extraction algorithm for path-following and landmark or goal recognition. Start-up information for the depth from motion algorithm could be provided as well, in addition to potential corners that are of use for localization purposes by the interest operator. Finally, if the robot becomes sufficiently disoriented relative to its global map, the schema interpretation system could be reinvoked to enable the robot to regain its bearings relative to the modeled world.

5. Modular Vision Algorithms

Although nothing in AuRA restricts sensor processing to be predominantly visual, much of the architecture is constructed to utilize this form of sensing. Action-oriented perception is the fundamental premise on which motor schema sensing and navigation is based. It is not necessary for the robot to fully understand the entire scene before navigation can be initiated (although this would certainly make things easier). Instead, by directing specific sensing strategies and the available computational resources to the motor needs of a particular task, only those portions of the scene which can contribute to the attainment of the pilot's goals are analyzed. Particular sensor algorithms are chosen to fit the demands of the specific path leg at hand.

No single perception algorithm is a panacea for navigation. The designer's goal instead is the development of a wealth of visual and other sensing algorithms which can provide the breadth that multi-domain navigation requires. AuRA should be able to provide navigational capabilities in both indoor and outdoor environments, allowing for considerable environmental diversity in each of these cases.

Computationally efficient vision algorithms are used to provide navigational information for the robot and are initially implemented on a single processor. The next stage will be to dedicate specific processors for each algorithm to improve performance and then to eventually distribute the load over parallel hardware.

From an experimental point of view, this architecture affords the flexibility to try new perceptual strategies without forcing significant changes in the supporting system components. By embedding motor actions as behaviors and perceptual strategies as focus of attention mechanisms, both represented in a schema form, the addition, modification and deletion of these program

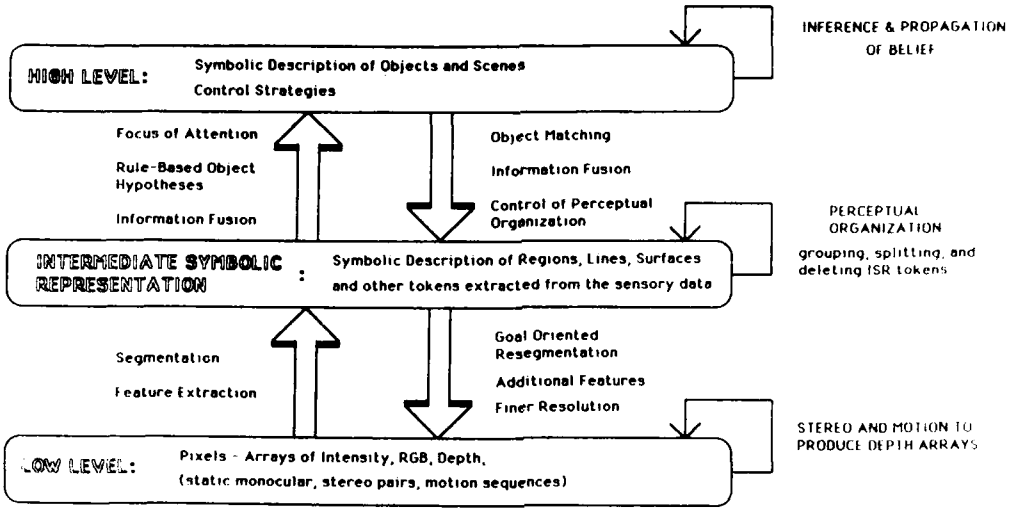


Figure 7. Multiple Levels of Representation and Processing in VISIONS

units is manageable. The emphasis is on modularity. New world representations can be embedded in LTM through the use of the feature editor, providing for extensions that may be needed by new algorithms.

A common thread running through many of the algorithms is their ability to be decomposed into two phases: start-up (bootstrap) and update (feedforward). The start-up phase performs more slowly and has less, if any, *a priori* knowledge to work from. The start-up process produces initial region seed statistics, depth information, line orientation, etc., which can be used to advantage in subsequent frame analysis. The update stage uses the information provided from the start-up phase to restrict the possible interpretation of image events and limit the search area for those events, thus reducing processing time significantly. The initial output of the start-up phase is updated after each processing run and is fed forward to provide a basis to guide analysis of the next image.

The remainder of this section will discuss some of the sensor algorithms that exist or are being developed for use within AuRA. The emphasis will be on visual processing, but some work already has been accomplished using ultrasonic data as well. The imaginative reader will undoubtedly think of other approaches and other sensors that can be used within a system such as AuRA. The strategies described below are not exhaustive, but rather they represent the current initial elements being introduced by VISIONS researchers for use within this framework.

5.1 Line Extraction

Line extraction has the potential for multiple uses within AuRA. These include path edge extraction for use by **stay-on-path** schemas, landmark identification for **find-landmark** schemas, and as a texture measure for terrain identification. Of these, the first two are currently being developed for use in AuRA. The remainder of this section will first describe the fast line finding algorithm, and then its application to both path following and localization purposes.

5.1.1 Fast Line Finder (FLF)

A fast line finder based on Burns' algorithm [15] has been developed by Kahn, Kitchen and Riseman. It is a two pass algorithm which first groups the image data based upon coarse quantization buckets of gradient orientation into edge-support regions. This grouping process collects pixels of similar gradient orientation into separate regions via a connected components algorithm. The gradient magnitude does not affect the line extraction process. A line is then fitted to the resultant edge support region. FLF differs from the original Burns' approach by simplifying the specification of the gradient orientation buckets and the extraction of the representative line for each edge support region. Many of the elementary computations can be sped up further through the use of a conventional pipeline processor which supports a look-up table and convolution processing.

Fragmentation of a potentially long image line often occurs if no *a priori* knowledge is available regarding the approximate orientation of the line in the image. The likelihood of extracting a particular long line increases by tuning the bucket's orientation to be centered on the anticipated orientation of a road edge or other line model in the image through the use of available knowledge extracted from LTM or previous images.

A key concept is *action-oriented perception*, performing only that computation which is necessary for the specific task at hand. Features available within the FLF algorithm to support this concept are described in the remainder of this paragraph. These features include the ability to scope the image (i.e. perform line extraction on a subwindow of the image). If the robot has an approximate knowledge of the position of the line feature being sought, (derived from LTM, the spatial error map and or previous images), substantial processing reductions can be attained by ignoring those portions of the image where the feature is unlikely to occur. In addition to orientation, the FLF can be adjusted to filter lines based on gradient magnitude, dispersion, size of the region, and length. By adjusting these filters in advance, based on the features desired (e.g. short lines for texture, or long lines for roads) unnecessary processing can be minimized. A secondary filtering procedure is also available for removing lines after the fast-line finder has been run, making it possible to collect different sets of lines with different characteristics with only a single run of the more time-consuming FLF. This is possible because when the lines are produced, statistics regarding each line are collected and stored with the endpoint data for later reference.

Figure 8a is an image of a sidewalk scene. Figure 8b shows the results of the FLF using the default bucket orientation for the entire image, and fig. 8c shows the results with the buckets tuned and scoped to the anticipated road edge based upon the internal

model of the vehicle position and orientation, while fig. 8d shows the results with the buckets tuned to horizontal and vertical edges, filtering to retain longer lines and with the image scoped above the horizon.

5.1.2 Path Following

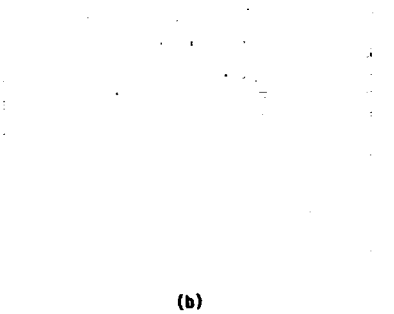
Using line following to extract path boundaries requires the grouping of resultant FLF line fragments into a single line representing each path edge. No effort is being made to condition or modify existing paths to make this process easier (e.g. by adding stripes, cleaning, etc.). The grouping strategy used must be able to deal with fragmentation and edge discontinuities, such as path intersections, leaves, etc.

If the uncertainty of the vehicle is within reasonable limits, predictions of the position and orientation of the road lines in the image plane can be made. As described above, there are two distinct components of road-following (see also [4]): the bootstrap or start-up phase, where the road edge is determined in the image for the first time; and the feedforward or update phase - where a previous image is used to guide the processing for the next image. Line finding is not necessarily the best strategy for initially finding the road's position. Nonetheless it can be reasonably effective if the road appears on a global map of the terrain and there is approximate information about the vehicle's position and orientation. These are both present within AuRA, in LTM and the spatial error map respectively.

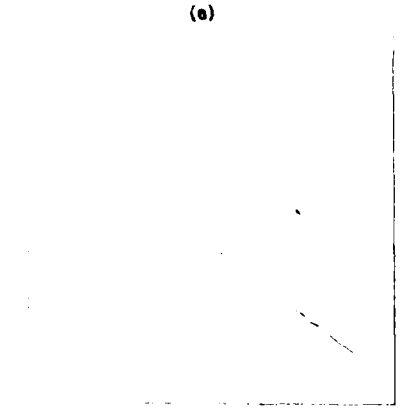
The feedforward phase assumes that the approximate position of the road was known in the last image. This information, when coupled with the commanded translation and rotation the robot has undertaken since the last image acquisition, can be used to predict where and at what orientation the road edges will occur in the newly acquired image. As anyone who has worked with mobile robots knows, the motion that a robot actually takes may differ quite significantly from that which it was commanded to perform. Consequently, there must be a considerable margin for error in these predictions if the algorithm is expected to be robust. Additionally, there must be some measure of the confidence in the line produced representing the road edge.

Path edge grouping proceeds as follows: The buckets are tuned based on the anticipated position of the road edge in feedforward mode; in bootstrap mode either LTM or the default buckets would be used. The fast line finder is then run, producing line fragments in the approximate orientation of the path edge (fig. 9a). These fragments are then filtered based on their distance from the anticipated image line and the expected orientation of either the right or left edge. Again the amount of tolerance allowed is controllable. This yields two sets of line fragments (one for each path edge - fig. 9b-c). All the fragments above the vanishing point of the road, (obtained from feedforward information), are discarded. The center of mass of the midpoints of remaining line fragments is computed, each midpoint weighted by the length of the fragments themselves. The average orientation is computed in a similar manner. The resulting point on the line and computed line orientation determine the line equation for each road edge. The left and right edges are then used to compute the road centerline (fig. 9d). The centerline is the basis for determining the rotational deviation of the vehicle relative to the road's vanishing point as well as the translational deviation from the road centerline. These newly

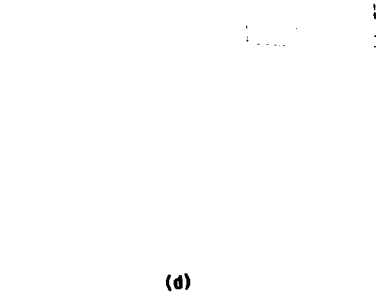
(a)



(b)



(c)



(d)

Figure 8. Fast Line Finding
a) Original sidewalk image.
b) Default bucket orientation.
c) Buckets tuned to road edges.
d) Buckets tuned to long vertical and horizontal lines above horizon.

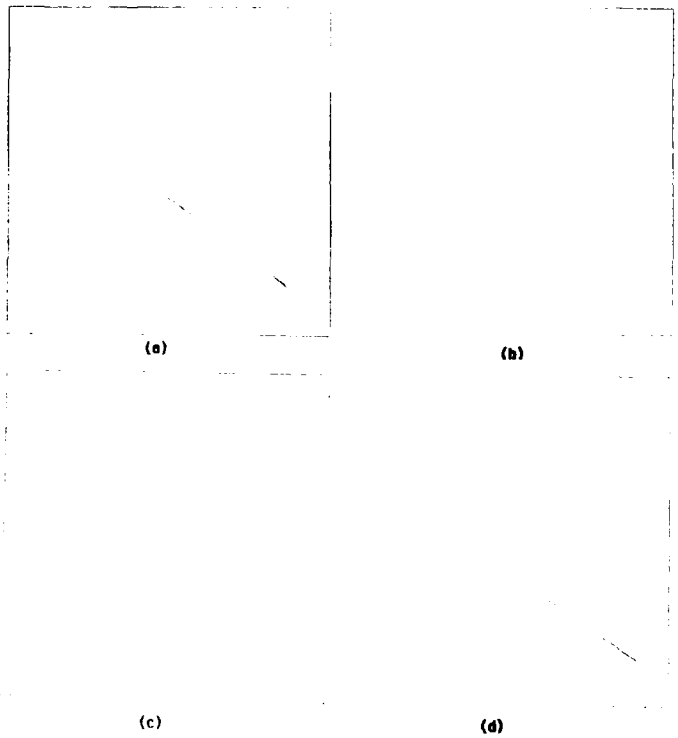


Figure 9. Path finding using FLF

a) Output of FLF when run on image 11a.
(tuned buckets - same as 11c)
b) Fragments left after filtering and windowing for left path edge.
c) Fragments left after filtering and windowing for right path edge.
d) Resultant path edges and computed road centerline.

computed path edges are then used as the models for the next feedforward step.

The total length of the line fragments used in producing the path edges serves as a measure of uncertainty. If this value drops below a specified threshold, special processing is undertaken. This includes increasing the error tolerances and margins in the FLF to see if a more confident line can be extracted from the same image, or if that fails, to digitize another image in the event that a passing obstacle blocked one or both path edges. If both of these strategies fail, the robot will reposition itself slightly and try another image. If this yet fails, alternate bootstrapping methods must be brought to bear.

The robot has already been able to successfully navigate both an outdoor sidewalk and an indoor hall using the FLF. Approximately 10 CPU seconds (VAX-750) are required for each step to provide the robot information for traveling 5.0 feet ahead. The 512 by 512 image digitized on a Gould IP8500 is averaged to 256 by 256 before line extraction. This time will be reduced by using pipelined hardware available on the digitizer. The vehicle servos on the computed center line position, correcting both orientation and translational drift as it proceeds.

5.1.3 Landmark Identification through Line Finding

Vehicle localization can be addressed by the line finding algorithm using data stored in LTM. Localization is simply orienting the vehicle relative to its global map; in other words getting its bearings. We do not propose that lines are the only mechanism for localization, but should serve in conjunction with other relevant algorithms. In the role of a confirmation mechanism, or

Figure 10. Road extraction via region segmentation
a) Sidewalk image.
b) Resultant extracted image representing road.
c) Resultant extracted image representing central portion of sky.

for tracking frame-to-frame a previously identified landmark feature, FLF localization is well suited. Extracting the edges of a path as described above also provides information for localizing the vehicle as well, assuming the path is represented in the world map.

Long, strong vertical lines and corners derived from such lines are probably the most appropriate general category of lines that is suitable for this application. Edges of buildings, telephone poles, lampposts or doorframes can be tracked using the line finder. Figure 8d shows the result of running the FLF on image fig. 8a with the buckets and filters tuned for long horizontal and vertical lines of high gradient magnitude. This orientation can be used to identify the roofs of buildings against the sky or road intersections directly in front of the vehicle. By windowing the image for a certain landmark based on the position of the vehicle as indicated by the spatial error map and *a priori* knowledge of the global coordinates and dimensions of the feature in question (from LTM), it becomes possible to isolate features such as the corner of a building by combining the evidence from both horizontal and vertical lines. This then can be used to constrain the positional error of the vehicle by backprojecting the 2-D data to 3-D world coordinates when combined with the knowledge of the height of the feature.

5.2 Fast Region Segmenter (FRS)

FRS is a region extraction algorithm developed in a manner akin to the fast line finder, but based upon similarity of color and intensity features. It functions by first defining a look-up table that is used for classifying an input image. The input image used can be an intensity image, a gradient image, a color plane, etc. This input image can be scoped (windowed) as is the case with the FLF. The look-up table maps ranges of pixel values to specific region labels and, as before, can be loaded in a top-down manner based upon stored knowledge or sampled data from previous images. Available knowledge can be used to define expected ranges of spectral attributes of interesting objects (fig. 7,10,11). The resulting classified image is then subjected to a region extraction algorithm which groups the classified pixels into regions. Statistical data is then collected regarding each region. This algorithm has been motivated by histogram-based segmentation algorithms [34], but achieves great simplification via constraints from stored object knowledge in LTM or the result of processing previous frames, to define the look-up table ranges for objects in the next frame.

The speed of this algorithm arises from the use of the look-up table to provide a quick mapping to the image. The connected components routine is then run on a restricted portion of the image selected through the use of top-down map constraints.

This segmentation can be used for road extraction as in [16,17]. Preliminary experimentation using intensity images can be seen in figure 10. Fig. 10a shows the original image and fig. 10b the region extracted representing the road. The statistics collected for the road region are then used for providing the expectations (feedforward) for the next image in the sequence [as in 16].

Landmark extraction can be handled similarly. The centroid of the landmark can be used for localization purposes, in contrast to the edge detection methods used by the FLF or the corner detection approach used with the Moravec operator described below. A bright yellow road sign (very dark in the blue sensory band) is segmented for localization purposes in figure 11.

5.3 Depth from Motion

Passive navigation by the determination of the position of environmental objects through vision is an important sensor strat-

Figure 11. Landmark identification via region segmentation
a) Original sign image (combined RGB intensity).
b) Blue plane of (a) chosen for analysis due to the spectral data of anticipated landmark (available from LTM).
c) Extracted region representing sign.

egy for AuRA. The motion research group within VISIONS has long explored the extraction of depth from motion [18,19,20]. A more recent algorithm, developed by Bharwani, Riseman and Hanson, uses a sequence of frames to incrementally refine positional estimates of objects over time. It can be used in mobile robotics for obstacle avoidance, position localization, and as evidence in object identification.

5.3.1 Algorithm

A brief sketch of the multiple frame depth-from-motion algorithm developed by Bharwani, Riseman, and Hanson follows. The reader is referred to [21,28] for the details of this approach. The algorithm allows refinement of depth over time up to some detectable limit, while maintaining a constant computational rate. This is very important for real-time processing.

The problem of recovering depth from motion in a sequence of images again involves the decomposition of the problem into two components: start-up and updating. This algorithm makes the assumption that the camera is undergoing pure translational motion and the position of the focus of expansion (FOE) is known within some reasonable estimated degree of accuracy. (These assumptions are not necessarily safe when using real world images. Frame registration and FOE recovery are problems that need to be solved. A discussion appears in sec. 5.3.2). This implies that the image displacement paths for a static environmental feature are constrained to move in a straight-line emanating radially from the FOE. An interest operator is used to extract points of high curvature and contrast in the image that are unlikely to correlate well with false match points in future frames. It is assumed that the obstacles or landmarks will exhibit some such points on their boundaries. This is probable if the backdrop is bland (e.g. the road itself) or by deliberately retrieving only "interesting" landmarks from LTM. However, it should be expected that interest points will be extracted from relevant and non-relevant image events.

The correspondence problem is the principal difficulty, how can one be sure that the feature in one frame corresponds to the same feature in the next frame after the robot has undergone translation. The start-up phase involves finding initial correct feature correspondences between the first two frames, while the update phase involves the use of the start-up analysis and the consequent approximate depth values to restrict the search area for corresponding matches at a higher match resolution in subsequent frames, thus reducing computation and providing refinements of the original depth estimates. Work by Snyder [22], addressing the limits of uncertainty in this type of motion, is fundamental to efficient use of previous correct correspondences in constraining the match in future frames.

Assuming the robot is traveling at a known velocity, the pixel displacements found between the first two images of a sequence (start up) can be used to further reduce the search for feature matching in successive frames, once the images have been registered so that non-translational motion of the camera has been subtracted out. Known sensor motion leads to a constraint on the match path and approximate depth (from start-up) constrains the portion of the path to be matched. Progressive refinements can be made in the estimation of feature displacement and hence distance to a relevant feature.

Different strategies such as histogramming the collection of points on the basis of depth, determining orientation of surfaces based upon the depth of several points on associated regions, or identifying landmarks by correlating distance from the viewer with the objects in the environmental map in LTM, can be used to extract objects from the environment. This data can then be used to provide information to the motor schema manager for effecting evasive action in the case of obstacles or for use in localization in the case of landmark location. The specific application of this technique to both of these cases appears below.

(a)

(b)

(c)

Figure 12. Depth from Motion Results
a-b) Two image sequence (distance traveled is 0.5 m)
c) Interest points tracked (from image 1)

The steps are at a distance of 13.5 meters in frame (a). The results for the interest points associated with the steps are within 10 percent of the ground truth.
(from [28]) (image sequence from CMU terrevator)

5.3.2 Applications

A primary goal of the depth-from-motion algorithm is to be able to provide information about the distance of an object lying in the path of the robot. In obstacle avoidance applications, computational requirements can be made tractable by restricting the processing to interest points (i.e. trackable image points of high contrast and curvature) and only to those that are lying within the current path of the robot.

Figure 12 illustrates some preliminary results of using the depth-from-motion algorithm for obstacle avoidance. It can be seen that the results, (accuracy within 10% in the recovery of obstacle depth), indeed look promising although continual research effort will be made to validate the initial results. The biggest problems encountered in the use of this algorithm in mobile robotics include first, accurate recovery of the FOE, which can be minimized through accurate calibration, and second, ensuring registration of the images. Stabilizing the camera with a gyroscopic platform affords a hardware solution to the registration problem. A software solution can be achieved by applying the correlation matching process to points near and above the horizon, i.e. distant (hence relatively unmoving) features that can be registered from frame to frame. The algorithm is quite sensitive to rotation so every effort should be made to minimize or eliminate any pitch, roll, or yaw movements of the camera relative to the scene.

The motion algorithm can be used for landmark identification as well. This is actually a simpler task than obstacle avoidance in many respects due to the availability of LTM knowledge to guide processing in a top-down manner. The approximate distance of a known landmark to the vehicle in a restricted portion of the overall image restricts the computation required substantially. When approximate ranges for the distance to an obstacle are known, the algorithm will perform more robustly than when underconstrained. Portions of the image can be searched that are outside of the obstacle avoidance regions. As these are usually further from the FOE than points in the robot's direction of motion, greater pixel displacements will occur and hence better results in the depth analysis. There is also a bottom-up strategy - obtain the depth of a subset of points that are on an anticipated object and the expected surface orientation from STM and search the depth points returned for a match. By overlapping region and line information, signs, buildings, telephone poles, etc., could be located within the robot's frame of reference.

5.4 Interest Operators

Interest operators are used in computer vision to pick out pixels associated with regions of high curvature and contrast. The Moravec operator [25] and the Kitchen-Rosenfeld gray-level corner detection interest operator [26] are two well-known examples. The depth from motion algorithm, described in section 5.3, uses an interest operator, (currently Moravec's), to determine the points on which to run the correspondence algorithms from frame to frame.

Interest operators are quite primitive as a stand-alone method for obtaining information for navigation. Their primary advantage is speed. By combining knowledge available from long-term memory with image data, it becomes possible to use interest operators to confirm the position of landmark corners. A clear-cut example would be the position of a building corner against the sky. When combined with knowledge from the robot's positional error map and available data from LTM, this method can be used in restricted circumstances to confirm the position of a real corner as predicted by the line-finding intersection method (sec. 5.1.3). A succinct description of the Moravec operator appears in [27] for those readers unfamiliar with its operation.

As the interest operator provides a measure of distinctiveness, (how different the pixel region is from its surroundings), the Moravec operator can also be used as a trigger event for spawning **avoid-obstacle** schema instantiations. When distinctive events occur against the relatively unchanging road backdrop, this would indicate a potential obstacle. This low-cost focus of attention mechanism permits the concentration of higher-cost computational effort in such likely sitations.

6. Summary

AuRA is a mobile robot system architecture that provides the flexibility and extensibility that is needed for an experimental testbed for robot navigation. By allowing for the incorporation of *a priori* knowledge in long-term memory, a variety of different perceptual strategies can be brought to bear by the robot in achieving its navigational goals. In particular, the individual motor schemas and their associated perceptual schemas can be added or deleted from the overall system without forcing a redesign.

A hierarchical planner determines the initial route as a sequence of legs to be completed over known terrain with predicted natural landmarks. Typical objects encountered in extended man-made domains (the interior of buildings, and outdoor settings with buildings and/or paths present) provide the information necessary for localization. The information gleaned from LTM is used to guide the pilot in the selection and parameterization of appropriate motor schemas and their associated perceptual schemas for instantiation in the motor schema manager. Actual piloting (sensor-driven navigation) is conducted by the motor schema manager. Positional updating occurs concurrently with the actual path traversal.

The vision algorithms to be used in AuRA encompass a wide range of computer vision techniques. These include primitive interest operators, more sophisticated line-finding and region segmentation algorithms, a multiple frame depth-from-motion algorithm, and a scene interpretation system. Each approach has its purpose, advantages and disadvantages for use in mobile robot navigation. In all cases, however, versions of vision algorithms have been developed which will extract image features rapidly (at the expense of reliability in some cases). In addition, the control of all algorithms attempts to use a top-down strategy of restricting processing to windows based upon LTM or previous frames. In this manner, real-time processing may be achieved for certain interesting navigation tasks that might not have been feasible until more powerful parallel hardware arrives.

Ongoing and future work includes the refinement of the individual perception algorithms used, increased real-time mobile robot experiments as parallel hardware is integrated into the system, and addressing the considerable system integration problems involved with the interfacing of the individual components being developed.

Acknowledgments

The authors would like to thank the members of the Laboratory for Perceptual Robotics and VISIONS groups for their assistance in the development of software and preparation of this paper. Special thanks are due to Raj Bharwani, Phil Kahn and Les Kitchen.

References

1. Parma, C., Hanson, A. and Riseman E., "Experiments in Schema-driven Interpretation of a Natural Scene", COINS Tech. Rep. 80-10, Comp. and Info. Sci. Dept., Univ. of Massachusetts, 1980.

2. Weymouth, T., "Using Object Descriptions in a Schema Network for Machine Vision", *Ph.D. Dissertation, COINS Tech. Rep.* 86-24, Univ. of Massachusetts, Amherst, May 1986.

3. Shen, H. and Signarowski, G., "A Knowledge Representation for Roving Robots", IEEE Second Conf. on Artif. Intel. Appl., pp. 621-628, December 1985.

4. Waxman, A.M., Le Moigne, J., and Srinivasan, B., "Visual Navigation of Roadways", Proc. IEEE Int. Conf. Robotics and Automation, St. Louis, Mo., pp. 862-867, 1985.

5. Arkin, R., "Internal Control of a Robot: An Endocrine Analogy", unpublished paper, Dec. 1984.

6. Arkin, R., "Path Planning for a Vision-based Mobile Robot", to appear in MOBILE ROBOTS - *SPIE Proc. Vol. 727*, 1986. Also COINS Technical Report 86-48, Dept. of Computer and Information Science, Univ. of Mass., 1986.

7. Arkin R., "Motor Schema Based Navigation for a Mobile Robot: An Approach to Programming by Behavior", to appear in Proc. IEEE International Conference on Robotics and Automation, Raleigh, N.C., 1987.

8. Arkin, R.C., Ph.D. Proposal, Computer and Information Science Department, University of Massachusetts, Spring 1986.

9. Crowley, J., "Navigation for an Intelligent Mobile Robot", CMU Robotics Institute Tech. Rep., CMU-RI-TR-84-18, 1984.

10. Chatila, R. and Laumond, J.P., "Position referencing and Consistent World Modeling for Mobile Robots", Proc. IEEE Int. Conf. Rob. and Auto., St. Louis, Mo., pp. 138-145, 1985.

11. Giralt, G., "Mobile Robots", Artificial Intelligence and Robotics, ed. Brady, Gerhardt, and Davidson, Springer-Verlag, NATO ASI series, pp. 365-394, 1984.

12. Giralt, G., Chatila, R., and Vaisset, M., "An Integrated Navigation and Motion Control System for Autonomous Multisensory Mobile Robots", *Robotics Research. The First International Symposium*, MIT Press, pp. 191-214, 1984.

13. Krogh, B., "A Generalized Potential Field Approach to Obstacle Avoidance Control", *SME - RI Technical Paper* MS84-484, 1984.

14. Krogh, B. and Thorpe, C., "Integrated Path Planning and Dynamic Steering Control for Autonomous Vehicle", *IEEE Conf. on Robotics and Auto.*, 1986 pp. 1664-1669.

15. Burns, J., Hanson, A., Riseman, E., "Extracting Straight Lines", Proc. IEEE 7th Int. Conf. on Pattern Recognition, Montreal, Can., pp. 482-485, 1984.

16. Thorpe, C. and Kanade, T., "Vision and Navigation for the CMU Navlab", to appear in MOBILE ROBOTS - *SPIE Proc. Vol. 727*, 1986.

17. Tou, J., " Software Architecture of Machine Vision for Roving Robots", Optical Engineering, Vol. 25-3, pp. 428-435, March 1986.

18. Williams, T., "Computer Interpretation of a Dynamic Image from a Moving Vehicle", COINS Technical Report 84-22, Dept. of Computer and Information Science, Univ. of Mass., May 1984.

19. Lawton, D., "Processing Dynamic Image Sequences from a Moving Sensor", Ph.D. dissertation, COINS Technical Report 84-05, Dept. of Computer and Information Science, Univ. of Mass., 1984.

20. Anandan, P., "Measuring Visual Motion from Image Sequences, Ph.D. Dissertation, University of Massachusetts at Amherst, Jan. 1987.

21. Bharwani, S., Riseman, E. and Hanson, A., "Refinement of Environmental Depth Maps over Multiple Frames", *Proc. IEEE Workshop on Motion Representation and Analysis*, pp. 73-80, May 1986.

22. Snyder, M., "The Accuracy of 3D Parameters in Correspondence-based Techniques", COINS Technical Report 86-28, Dept. of Computer and Information Science, Univ. of Mass., July 1986.

23. Hanson, A. and Riseman, E., "VISIONS, A Computer System for Interpreting Scenes", COMPUTER VISION SYSTEMS (Hanson and Riseman eds.), Academic Press, pp. 303-333, 1978.

24. Hanson, A., and Riseman, E., "A Summary of Image Understanding Research at the University of Massachusetts", COINS Tech. Report 83-35, Dept. of Computer and Information Science, Univ. of Mass., 1983.

25. Moravec, H., *Robot Rover Visual Navigation*, UMI Press, 1981.

26. Kitchen, L. and Rosenfeld, L., "Grey-level 'orner Detection", Technical Report 887, Comp. Sci. Ctr., U. Maryland, College Park, Md. 1980.

27. Ballard, D. and Brown, C., COMPUTER VISION, Prentice-Hall, 1982, pp. 69-70.

28. Bharwani, S., Riseman, E. and Hanson, A.,"Multiframe Computation of Accurate Depth Maps using Uncertainty Analysis", COINS Tech. Report (in preparation), Dept. of Computer and Information Science, Univ. of Mass., 1986.

29. Hanson A. and E. Riseman, E., "The VISIONS Image Understanding System - 1986", in ADVANCES IN COMPUTER VISION, (Chris Brown ed.), to be published by Erlbaum Press.

30. Weems, C., "Image Processing on a Content Addressable Array Parallel Processor", Ph.D. Dissertation and COINS Tech. Report 84-14, Dept. of Computer and Information Science, Univ. of Mass., Sept. 1984.

31. Draper, B., Hanson, A. and Riseman ,E., "A Software Environment for High Level Vision", COINS Tech. Report (in preparation), Dept. of Computer and Information Science, Univ. of Mass., 1986.

32. Parodi, A.M., "Multi-Goal Real-time Global Path Planning for an Autonomous Land Vehicle using a High-speed Graph Search Processor", Proc. IEEE Int. Conf. Robotics and Automation, St. Louis, Mo., pp. 161-167, 1985.

33. Arbib, M., "Perceptual Structures and Distributed Motor Control", HANDBOOK OF PHYSIOLOGY - The Nervous System II (V. Brooks ed.), pp. 1449-1465,1981.

34. Kohler, R., "Integrating Non-semantic Knowledge into Image Segmentation Processes", Ph.D. Thesis, COINS TR-84-04, Dept. of Computer and Information Science, Univ. of Mass., 1984.

Guiding an Autonomous Land Vehicle Using Knowledge-Based Landmark Recognition

Hatem Nasr, Bir Bhanu and Stephanie Schaffer

Honeywell Systems and Research Center
3660 Technology Drive, Minneapolis, MN 55418

ABSTRACT

In the Autonomous Land Vehicle (ALV) application scenario, a significant amount of positional error is accumulated in the land navigation system after traversing long distances. Landmark recognition can be used to update the land navigation system by recognizing the observed objects in the scene and associating them with the specific landmarks in the geographic map knowledge-base. In this paper we present a novel landmark recognition technique based on a perception-reasoning-action and expectation paradigm of an intelligent agent. It uses extensive map and domain dependent knowledge in a model-based approach. It performs spatial reasoning by using N-ary relations in combination with negative and positive evidences. Since it can predicts the appearance and disappearance of objects, it reduces the computational complexity and uncertainty in labeling objects. It provides a flexible and modular computational framework for abstracting image information and modeling objects in heterogeneous representations. We present examples using real ALV images.

I. INTRODUCTION

In order to accomplish missions such as surveillance, search and rescue and munitions deployment, an Autonomous Land Vehicle (ALV) has to travel long distances. This results in a significant amount of positional error in the land navigation system. Landmark recognition is used to update the land navigation system, thus guiding the ALV to remain on its proper course. Landmarks of interest include telephone poles, storage tanks, buildings, houses, gates, etc.

Model-based vision has been a popular paradigm in computer vision since it reduces the problem complexity and no learning is involved. Binford [6] has given a summary of model-based vision work. He has described several systems including the work of Brooks [7] on ACRONYM, Riseman and Hanson's [12] work on VISIONS, and Nagao and Matsuyama's [14] work on the analysis of complex aerial photographs. McKeown et al [13] have used map and domain specific knowledge in the SPAM rule-based systems for the interpretation of airport scenes in aerial images. Hwang [10] has also used domain knowledge to guide interpretation of suburban house scenes in aerial imagery. He has used test-hypothesize-act sequence to generate large number of hypotheses which are then integrated into a consistent interpretation. Bhanu [1-4] has used several modeling and relaxation matching techniques for the recognition of 2-D and 3-D nonoccluded and occluded objects. As compared to all the previous related work, as mentioned in the above, the paradigm of an intelligent agent (like the ALV) which we have used here is based on the perception-reasoning-action and expectation cycle. Thus we have an expectation-driven, knowledge-based landmark recognition system called PREACTE (Perception-REasoning-ACTion and Expectation), that utilizes a priori, map and perceptual knowledge, spatial reasoning and knowledge aggregation methods. In contrast to the work of Davis [9], explicit knowledge about the map and landmarks is assumed to be given and it is represented in a relational network. It is used to generate an Expected Site Model (ESM) given the ALV location and its velocity. Landmarks at a particular map site have their 3-D models stored in heterogeneous representations. The vision system generates a 2-D and partial 3-D scene model from the observed scene. The ESM hypothesis is verified by matching it to the image model. The matching problem is solved by using object grouping and spatial reasoning. Positive as well as negative evidences are used to verify the existence of each landmark in the scene. The system also provides feedback control to the low-level processes to permit adaptation of the feature detection algorithms parameters to changing illumination and environmental conditions.

In the following, we present the details of the PREACTE system and examples of landmark recognition using real ALV imagery. It is worth mentioning that PREACTE is a component subsystem of a much larger system. Other parts of the system will be referred to but not described.

II. CONCEPTUAL APPROACH

The task of visual landmark recognition in the autonomous vehicle scenario can be categorized as (a) uninformed and (b) informed. In the uninformed case, given a map representation, the vision system attempts to attach specific landmark labels to segmented image regions of an arbitrary observed scene and infers the location of the vehicle in the map (world). On the other hand, in the informed case, while the task is the same as earlier, there is a priori knowledge (with a certain level of certainty) of the past location of vehicle in the map and its velocity. It is the informed case that is of interest to the discussion of this paper. There are a number of assumptions made in this landmark recognition approach. They include: 1) a forward looking fixed camera model is given, 2) traversal by the vehicle is allowed only on defined routes, and 3) minor range variations from a given site does not lead to major changes in the objects' appearance in the image and their spatial distributions.

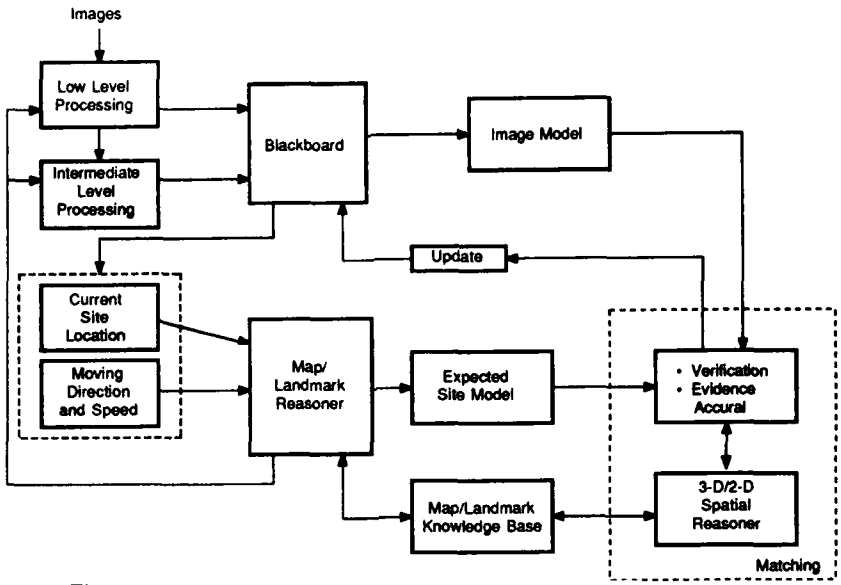


Fig. 1. PREACTE's top-level approach to landmark recognition.

Fig. 1 illustrates the overall approach for PREACTE's landmark recognition task. It is a top-down expectation-driven approach, whereby an Expected Site Model (ESM) of the map is generated based on domain-dependent knowledge of the current (or projected) location of the vehicle in the map and vehicle's velocity. The ESM contains models of the expected map site and its landmarks. This expectation provides the hypotheses to be verified by the content of an image to be acquired after a computed time t, given the velocity of the vehicle and the distance between the current site and the predicted one. While this approach may seem similar to other hypothesis-verification concepts, it is not only unique by its added expectations but also in its extensive and explicit domain specific knowledge which contributes to enhanced performance. Site models introduce spatial constraints on the locations and distributions of landmarks, by using a "road" model as a reference. Spatial constraints greatly reduce the search space while attempting to find a correspondence between the image regions and a model. This mapping is usually many-to-one in complex outdoor scenes, because of imperfect segmentation.

In the segmented image each region-based feature such as size, texture, color, etc. provides an independent evidence for the existence of an expected landmark. Evidence accrual is accomplished by an extension of a heuristic Bayesian formula [8], which will be discussed in Section II.3. The heuristic formula is used to compute the certainty about a map site location based on the certainty of the previous site and the evidences of each landmark existence at the current site. Similar formulation was suggested by Lowe [11] for evidential reasoning for visual recognition.

A. Map/Landmark Knowledge-Base

Extensive map knowledge and landmarks models are fundamental to the recognition task. Our map representation relies heavily on declarative and explicit knowledge instead of procedural methods on relational databases [13]. The map knowledge is represented in a hierarchical relational network, as illustrated in Fig. 2. The entire map is divided into 25 sectors (5 horizontally and 5 vertically). Each sector contains four quadrants which in turn contain a number of surveyed sites (Fig. 3). All map primitives are represented in a schema structure. The map dimensions are characterized by their cartographic coordinates. Schema representation provides an object-oriented computational environment which supports the inheritance of different map primitives properties and allows modular and flexible means for searching and updating the map knowledge base. The map sites between which the vehicle traverses have been surveyed and characterized by site numbers. An aerial photograph with numbered sites is shown in Fig. 3. Knowledge acquired about these sites includes: approximate < latitude, longitude, elevation >, distance between sites, terrain descriptions, landmarks labels contained in a site, etc. Such site information is represented in a SITE schema, with corresponding slots, as illustrated in Fig. 2. Slots names include: HAS_LANDMARKS, NEXT_SITE, LOCATION, SPATIAL_MODEL, etc. A critical slot is NEXT_SITE which has an "active" value. By active, it is meant that it is dependent on a variable (demon) which is the vehicle direction (North, South, etc.). For different traversal directions from the current site, the names of the neighboring sites are explicitly declared in the NEXT-SITE slot, as shown in Fig. 4. The SPATIAL_MODEL defines the "expectation zone" of the landmarks (in the image) with respect to the road and with respect to each others. It also specifies the minimum and maximum distance of each landmark from the road borderline. Each landmark is represented as a schema or a collection of schemas. Each landmark is represented as an instance of a landmark-class which, in turn, is an instance of an object-class. For example, T-POLE-17 is an instance of POLE, which is an instance of MAN_MADE_OBJECTS. Instances in this case inherit some properties and declare others.

This declarative and hierarchical representation of the knowledge allows not only a natural conceptual mapping, but also a flexible means for pattern matching, data access, tracing of the reasoning process and maintenance of the knowledge base. The slots and their values in a LANDMARK schema correspond to the landmark's attributes such as color, texture, shape, geometric model, etc. The landmark attributes are characterized

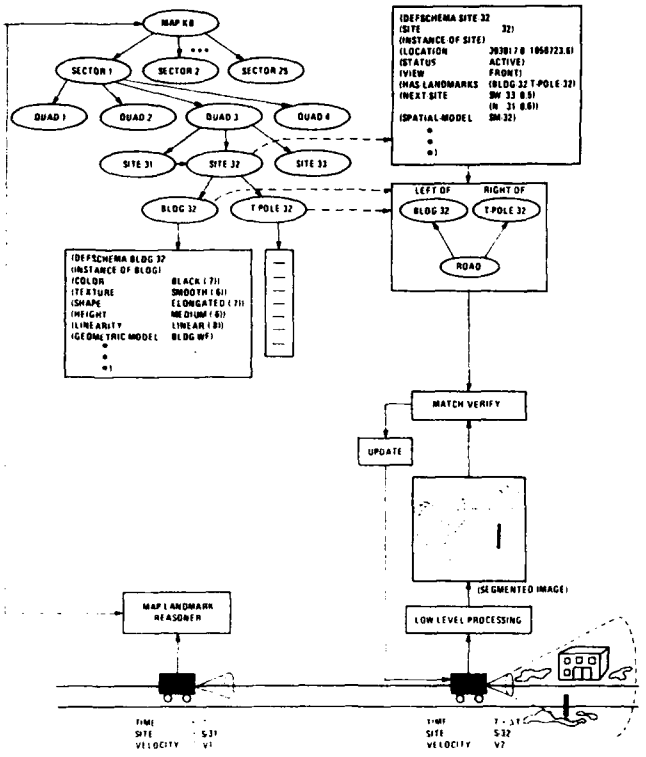


Fig. 2. Map knowledge representation and graphic illustration of the approach based on the perception-reasoning-action and expectation paradigm.

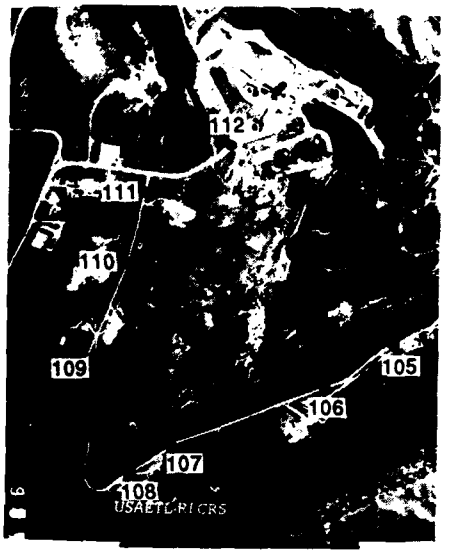


Fig. 3. Aerial photograph of the map.

symbolically, such as color is "black", texture is" smooth", and shape is "elongated". Each attribute's value is assigned a likelihood that characterizes its discriminant strength. For example, the fact that poles are elongated, place a high likelihood value (0.8) on having an elongated shape. The knowledge acquisition for modeling each landmark in the knowledge base is performed by abstracting and characterizing map data through actual observations and measurements, and observations of images taken at individual sites. The groundtruth values of each landmark attribute are obtained from a combination of actual metrics, and approximations of features extracted from hand segmented images. Three dimensional geometric models are represented in the geometric-model slot of a LANDMARK schema. Different modeling techniques are available for different landmarks. For example, buildings are represented as wire-frames, while poles are represented as generalized cylinders. Thus models are allowed to have heterogeneous representations. Image description is obtained by projecting the 3-D model on a 2-D plane using perspective transformations. This hybrid representational framework for object modeling provides a unique ability to combine different types of object descriptions (semantic, geometric and relational). This in turn allows the system to perform more robustly and efficiently, and recover from a single bad representation.

B. Prediction

Given the a priori knowledge of the vehicle's current location in the map space and its velocity, it is possible to predict the upcoming site that will be traversed through the explicit representation of the map knowledge and the proper control procedures. The Map/Landmark Reasoner (MLR) provides such control, by invoking the active values in the NEXT_SITE slot of the current SITE schema, as described earlier. The ESM is a "provision" by the MLR to make the expected site and its corresponding landmark schemas as an "active" hypothesis to be verified. In parallel to predicting the next site, the distance between the current and the expected site along with the vehicle velocity are used to predict the (arrival) time at which the sequence of images should be processed to verify the hypothesized ESM. Evidence accrual and knowledge aggregation is dynamically performed between sites to confirm arrival time at the predicted site [5]. The ESM of SITE-110 shown in Fig. 7 is as follows:

```
(DEFSCHEMA  SITE-110
    (SITE                110)
    (LOCATION            (392961.7        1050742.9))
    (INSTANCE - OF       SITE)
    (STATUS              ACTIVE)
    (VIEW                FRONT)
    (HAS-LANDMARKS       (T-POLE-110 G-TANK-10 BLDG-110))
    (NEXT-SITE           (E 109  0.255)  (W 111  0.153))
    (SPATIAL-MODEL       SM-110)
    (TERRAIN-TYPE        NIL))

(SETQ     SM-110      '( (T-POLE-110    G-TANK-110     BLDG-110)
                         (T-POLE-110                   (LEFT-OF  ROAD)
                                                       (MIN-L-DIST-ROAD 160)
                                                       (MAX-L-DIST-ROAD 200)

                         (G-TANK-110                   (RIGHT-OF ROAD)
                                                       (MIN-R-DIST      200)
                                                       (MAX-R-DIST      250))

                         (BLDG-110                     (ABOVE      G-TANK-110)
                                                       (RIGHT-OF        ROAD )
                                                       (MIN-R-DIST      230)
                                                       (MAX-R-DIST      280) ) ) )
```

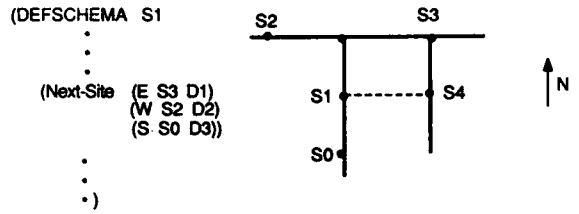


Fig. 4. Next-Site slot representation in a SITE schema provides the expected site and distance to it based on the vehicle direction.

Predictions are also used by the low-level image processing . A priori knowledge of the objects' attributes which will appear in the image and their relative locations guide the segmentation. A rule-based system is invoked to interpret the corresponding information in the ESM, which results in properly adapting the segmentation parameters based on the landmarks' distinguishing attributes, such as color , location, texture,etc.

C. Image Modeling

An image model for the task of landmark recognition is a collection of regions-of-interest extracted by a region-based segmentation method. A region-of-interest for this task is constrained by an upper and lower bound on its size. This means that after performing some region splitting and merging, most of the very small and the very large regions are merged or discarded from the image. In addition, regions-of-interest do not include any regions that represent moving objects in the scene as determined by the motion analysis module. A number of image features are extracted for each region, such as color, length, size, perimeter, texture, Minimum Bounding Rectangle (MBR), etc., as well as some derived features such as elongation, linearity, compactness, etc. All image information is available in the blackboard (Fig. 1), which is a superset model of all the results collected from different image understanding modules. The landmark recognition system operates as a "knowledge source". There are other knowledge sources with other tasks such as object recognition (other than landmarks) and motion analysis. The blackboard plays the role of a central knowledge structure among these different knowledge sources. During the process of extracting regions-of-interest for landmark recognition there is a risk of ignoring regions in the image that are part of the actual landmarks. These regions could have been split to very small regions or merged with very large ones, which is a natural outcome of the inherently weak segmentation methods. Symbolic feature extraction is performed on the region-based features. So, instead of having area = 1500 (pixels) and intensity = 52, we could have area = large and intensity = low. The symbolic characterization of the features using "relative" image information provides a better abstraction of the image and a framework for knowledge-based reasoning. On one hand, this has the advantage of making the feature space smaller, therefore easier to manipulate. On the other hand, it makes it insensitive to feature variations in the image.

Each set of region features is represented in a schema structure instead of a feature vector as in pattern recognition. This schema representation of regions does not have any conceptual justifications, however it provides a compatible data structure with the landmark models in the knowledge-base. Most of the region features have representative attributes in the landmarks models. This allows symbolic pattern matching to be performed easily by the high-level vision knowledge sources. Beyond that, it makes the reasoning process more traceable.

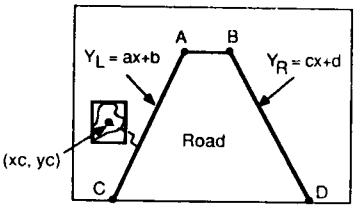


Fig. 5. Road model representation.

A critical region in the image is the road region, which is used as a reference in the image model. Spatial constraints are applied on the regions-of-interest to find which regions in the image fall to the left and to the right of the road. The road is easily segmented out in similar imaging scenarios, using current state-of-the-art road segmentation techniques (currently used in the ALV). This is assuming that it is a "structured" road (i.e., asphalt, concrete, etc.) that provides good contrast (not dirt roads). The road is represented in the model by its vertices and the approximate straight lines of the left and right borders, as shown in Fig. 5. For each region, we determine the position of its centroid and compute the shortest distance from the region to the road border line. This distance is compared to the constraint imposed on each landmark by the site spatial model. Thus we obtain the following top-level structure for the image model:

(<IM-#> <frame-#> <road-region-tag>
<number-of-regions-of-interest>
((<road-vertices>) (<left-border>) (<right-border>))
((<left-regions-list>) (<right-regions-list>))))

D. Hypothesis Verification

Given the expected site model (ESM) and the current image model, the objectives of the matching and verification process are two fold: (a) to label the regions in the image corresponding to the expected landmarks, and (b) to determine the certainty level of the predicted map site location. The process by which the first objective is accomplished is as follows: 1. find the set of regions {R} in the image model (IM) which satisfy the spatial constraints SC_i imposed by landmark l_i in the ESM SPATIAL_MODEL. This constraint application yields to more than one corresponding region r_j. 2. Compute the evidence $E(l_i)$ that each r_j in {R} yields, using the FIND_EVIDENCE algorithm. 3. The r_j that results in $E(l_i)_{max}$ (provided it is a positive evidence) is considered as best match candidate for l_i (there may be more than one given that their values surpass a certain threshold) . The second objective is achieved by aggregating the individual set of evidences ($E(l_i)_{max}$) and the certainty level about the previous map site location and the potential error introduced by range and the view angle of the camera.

The FIND_EVIDENCE algorithm considers that each landmark l_i in the ESM has a set of attributes $\{A_{i1}, ..., A_{iK}, ..., A_{in}\}$, each with a likelihood LH_{ik}, as described earlier. Each region r_j in {R} has a set of features $\{f_{j1}, ..., f_{jk}, ..., f_{jn}\}$. Note that A_{ik} and f_{ik} correspond to the same thing (in the model and the image), such as color, size, texture, etc. Given these features, we want to compute the evidence that l_i is present in the image.

$$P(l_i/f_{j1}, \ldots, f_{jk}, \ldots, f_{jn}) = \frac{P(l_i) * P(f_{j1}, \ldots, f_{jk}, \ldots, f_{jn}/l_i)}{P(f_{j1}, \ldots, f_{jk}, \ldots, f_{jn})} \quad (1)$$

By making the independence assumption among features in a region and among features occurrence in images, the above equation can be rewritten as:

$$P(l_i/f_{j1}, \ldots, f_{jk}, \ldots, f_{jn}) = \frac{P(l_i)*P(f_{j1}/l_i)*\ldots*P(f_{jk}/l_i)*\ldots*P(f_{jn}/l_i)}{P(f_{j1})*\ldots*P(f_{jk})*\ldots*P(f_{jn})}$$

$$= P(l_i) * \prod_{k=1}^{n} \frac{P(f_{jk}/l_i)}{P(f_{jk})} \quad (2)$$

where n is the number of features, $P(l_i)$ is the initial probability of a landmark being found in a given site. For now this is set to 1 for all landmarks. However, $P(l_i)$ is actually a function of the certainty level about the previous map site location, navigational error and other variables. $P(f_{jk})$ is the probability of occurrence of a feature in an image, which is equal to 1/(number of possible feature values). For example, if texture can take either of the four values: coarse, smooth, regular or irregular, then P (texture = smooth) = 1/4. Finally,

$$P(f_{jk}/l_i) = \begin{cases} LH_{ik} & \text{if } f_{jk} = A_{ik} \\ \dfrac{1 - LH_{ik}}{d(f_{jk}, A_{ik})} & \text{if } f_{jk} \neq A_{ik} \end{cases} \quad (2.1)$$

which is best explained through the following example:

Given two regions r_1 and r_2 in the image with different sizes (f_{jk}), SIZE (r_1) = SMALL and SIZE (r_2) = LARGE. Given a model of landmark L, with the expected size to be LARGE (A_{ik}), with a likelihood (LH_{ik}) of 0.7. The SIZE feature can take any of the following ordered values: {SMALL, MEDIUM, LARGE}. If r_2 is being matched to L, (2.1) yields to 0.7, because $f_{jk} = A_{ik}$. On the other hand, if r_1 is being matched to L, then (2.1) yields to (1-0.7)/2. The denominator 2 is used because LARGE is two unit distances (denoted by d(.)) from SMALL.We rewrite (2) as:

$$P(l_i/f_{j1}, \ldots, f_{jk}, \ldots, f_{jn}) = P(l_i) * \prod_{k=1}^{n} I(f_{jk}/l_i) \quad (3)$$

where I(./.) is the term within the product sign. The value of $I(f_{jk}/l_i)$ can be greater than 1, because the heuristic nature of the formulation does not reflect a probabilistic set of conditional events, as formulated in Bayes theory. Moreover, $P(l_i/f_{ji}...f_{jn})$ can result in a very large number or a very small positive number.

By taking the logarithm of both sides of (3), introducing W_j as a normalization factor for each feature, and dividing by the number of features (n), we have :

$$\text{Log}[P(l_i/f_{j1}, \ldots, f_{jk}, \ldots, f_{jn})] = \text{Log}[P(l_i)] + \frac{\sum_{k=1}^{n} \text{Log}[I(f_{jk}/l_i) * W_k]}{n} \quad (4)$$

where W_k is a normalization factor between 0 and 1.

Here we further simplify (4) and introduce the evidence terms E and e to be the logarithm of P and I*W respectively . So, the evidence formula can be written as follows:

$$E(l_i) = \frac{\sum_{k=1}^{n} e(f_{jk}/l_i)}{n} \quad (5)$$

The values of $E(l_i)$ fall between 0 and 1. If $E(l_i) > 0.6$ it means a "positive" set of evidences. On the other hand, if $E(l_i) < 0.3$ it is interpreted as "negative" evidence. Otherwise, $E(l_i)$ is characterized as "neutral".

An important characteristic of the PREACTE system is that it utilizes negative as well as positive evidences to verify its expectations. There are many types of negative evidences that could be encountered during the hypothesis generation and verification process. The one that is of particular interest to us is when there is a negative evidence about a "single" landmark ($E(l_i)_{max} < 0.3$) in conjunction with positive evidences about the other landmarks (average evidence >0.6) and a reasonable level of certainty about the previous site ($U_{S-1} < 5$) (discussed later). This case is interpreted as caused by one or more of the following: (a) error in the dimension of the expectation zone, (b) bad segmentation results, and (c) change in the expected view angle or range.

In such a case, we perform the following steps: 1. enlarge the expectation zone by a fixed margin, and find the evidences introduced by the new set of regions, as shown in Fig. 6. 2. If step 1 fails to produce an admissible set of evidences, then the expectation zone of the image is resegmented using a new set of parameters that are strictly object dependent.

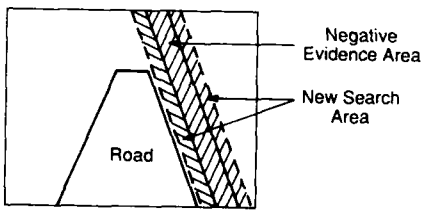


Fig. 6. New search area as a result of negative evidences.

Even though landmark recognition is introduced to assist the autonomous vehicle land navigation system, there is obviously uncertainty attached to the results of the recognition system. We compute the uncertainty U_S at each site location in the following manner:

$$U_s = (U_{s-1} + \alpha) * \prod_{i=1}^{m} \frac{0.5}{E(l_i)_{max}}$$

where U_{S-1} is the uncertainty at the previous site, U_0, the initial uncertainty, is equal to 1, a is the error factor introduced by the navigation system, it is set to a constant of 0.3 (for experimental reasons), and $E(l_i)_{max}$ is the maximum evidence of l_i. If two or more regions return evidences greater than .8 then the average is computed.The value 0.5 is used (as neutral evidence) to stabilize the function, m is the number of landmarks. The multiplicative nature of U_S provides it with the capability of rapidly recovering

its value given a high set of evidences at the current site and a high level of uncertainty at the previous site.

III. RESULTS

We have implemented a prototype system written in Common Lisp and ART (Automated Reasoning Tool) on the Symbolics 3670. The image processing software was implemented in C on the VAX 11/750. The Symbolics hosts all the high-level (symbolic) processing software, including the blackboard. The map and landmarks knowledge-base is implemented as a hierarchical relational network of schemas, the FIND-EVIDENCE algorithm is implemented in Lisp. The Map-Landmark Reasoner is implemented in a rule-based structure. An initial implementation of PREACTE was tested on a video sequence of imagery. Data was collected at 30 frames/second by a camera installed on top of a vehicle and driven on the road connecting the sites shown in Fig. 3.

The system was easily capable of predicting the next site and approximate arrival time at each site. In an experiment where we started at SITE-109 and traveled west at 10 kph, arrival time was predicted to SITE-110, at 245.4 seconds (distance between the two sites is .426 mile). Fig. 7(a) shows the image taken at SITE-110, which contains a pole (T-POLE-110) to the left of the road, a gas tank (G-TANK-110) and a building (BLDG-110) to the right. Initial segmentation of the image is shown in Fig. 7(b). As a result of region splitting and merging and discarding small regions, we obtain the image shown in Fig. 7(c). Rectangular boxes are overlayed over the regions recognized by PREACTE as landmarks, based on the hypotheses generated in the ESM of SITE-110 (shown in Section II.2). The hypothesis verification results are shown in the "match-evidence" column of Table I which contains a listing of the regions yielding the highest evidences and a subset of their features (other features include compactness, intensity variance, etc.), as represented in the image model. The spatial constraint specified by the spatial model (SM-110) yielded to a small number of regions-of-interest for the POLE hypothesis, as a result of the successful post-segmentation effort. More regions-of-interest were considered as candidates for the other landmarks. The road (region 98) in the image is modeled by its approximate left border ($y = -0.8x + 357.0$) and its right border ($y = 1.3x - 126.7$). The T-POLE-110 hypothesis produced two regions with high evidences. Since a threshold of 0.8 was used, both regions 30 and 79 are recognized as T-POLE-110. The lower part of the pole (region 79) is merged with some ground and background regions; nevertheless it still resulted in a higher evidence than the upper part (region 30). Currently effort is underway to implement a region grouping technique based on evidences, proximity, size and other criteria. The lower part of the tank was broken up into six small regions because of the illumination and shape factors. These regions were included in the hypothesis verification and they produced significantly lower evidences.The uncertainty :

$U_{110} = (1+0.3)*((0.5)^3/(0.875+0.62+0.70)) = 0.43$, where 0.875 is the average of 0.92 and 0.83.

IV. CONCLUSIONS

In this paper we have presented concepts and initial results of our perception-reasoning-action and expectation paradigm of our ongoing research for guiding the ALV by recognizing landmarks along the sides of the road. In the future, we will extend to a more general and complex situation where the ALV may be traveling through terrain and it has to determine precisely where it is on the map by using landmark recognition.

Fig. 7(a) Image obtained at site 110.

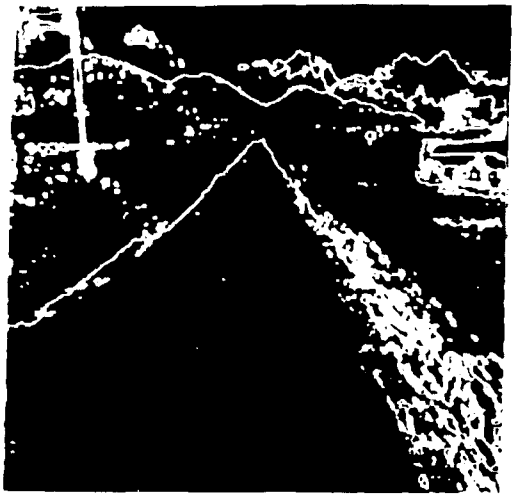


Fig. 7(c) Initial segmentation results.

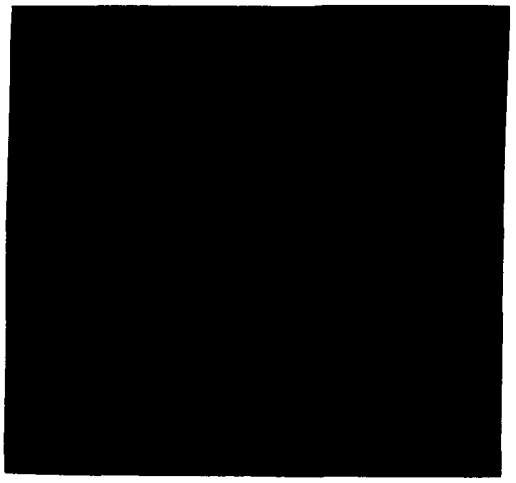


Fig. 7(c) Segmented image after region splitting, merging and discarding small regions. Regions highlighted by rectangles indicate the result of landmark recognition by PREACTE.

	Features							Recognition Results	
Region	Size	Color	MBR	Texture	Elongation	Shape	Location	Match-Evidence	Landmark Hypothesis
30	Small (91)	Black (25.3)	(99, 103, 61, 111)	Smooth	High (50:4)	Long and Linear	(101.2, 82.9)	0.83	T-POLE-110
79	Small (104)	Black (27.3)	(104, 105, 112, 132)	Irregular	High (20:1)	Long and Linear	(104.6, 121.9)	0.92	T-POLE-110
95	Medium (675)	White (224.5)	(445, 510, 140, 155)	Smooth	Low (05:15)	Not convex	(482, 148.3)	0.62	G-TANK-110
70	Medium (672)	Gray (199.7)	(469, 510, 99, 127)	Irregular	Low (41:28)	Linear	(490.2, 115. 3)	0.71	BLDG-110

Table I. Landmark recognition results.

References

[1] B. Bhanu, "Recognition of Occluded Objects," Proc. of the 8th International Joint Conference on Artificial Intelligence, IJCAI-83, Karlsruhe, West Germany, August 8-12, 1983, pp. 1136-1138.

[2] B. Bhanu, "Representation and Shape Matching of 3-D Objects," IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. PAMI-6, May 1984, pp. 340-351.

[3] B. Bhanu and O.D. Faugeras," Shape Matching of Two-dimensional Objects," IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. PAMI-6, March 1984, pp. 137-156.

[4] B. Bhanu and T. Henderson," CAGD Based 3-D Vision," IEEE International Conference on Robotics and Automation, March 1985, pp. 411-417.

[5] B. Bhanu and W. Burger, "DRIVE: Dynamic Reasoning Using Integrated Visual Evidences," Proc. DARPA Image Understanding Workshop, University of Southern California, Feb. 1987.

[6] T.O. Binford, "Survey of Model-Based Image Analysis," The International Journal of Robotics Research, Vol. 1, Spring 1982, pp. 18-64.

[7] R. A. Brooks, "Symbolic Reasoning Among 3-D Models and 2-D Images," Artificial Intelligence, Vol. 17, pp. 285-348, 1981.

[8] E. Charniak, "The Bayesian Basis of Common Sense Reasoning in Medical Diagnosis," Proc. American Association of Artificial Intelligence Conference 1983, AAAI-83, pp. 70-73.

[9] E. Davis, Representing and Acquiring Geographic Knowledge, Morgan Kaufman Publishers, Inc. 1986.

[10] S.V. Hwang, "Evidence Accumulation for Spatial Reasoning in Aerial Image Understanding," Ph.D. Thesis, Dept. of Computer Science, University of Maryland, College Park, Maryland, 1984.

[11] D. Lowe, Perceptual Organization and Visual Recognition, Kluwer Publishing Co., 1985.

[12] A.R. Hanson and E.M. Riseman, "VISIONS: A Computer System for Interpreting Scenes," in Computer Vision Systems, A.R. Hanson and E.M. Riseman, (Eds.), New York, Academic Press, 1978, pp. 303-333.

[13] D.M. McKeown, Jr., W.A. Harvey, Jr., and J. McDermott, "Rule-Based Interpretation of Aerial Imagery," IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. PAMI-7, September 1985, pp. 570-585.

[14] M. Nagao and T. Matsuyama, A Structural Analysis of Complex Aerial Photographs, Plenum Press, 1980.

The CMU Navigational Architecture

Anthony Stentz

Yoshimasa Goto

The Robotics Institute

Carnegie-Mellon University

Pittsburgh, PA 15213

Abstract

This paper describes the current status of the Autonomous Land Vehicle research at Carnegie-Mellon University's Robotics Institute, focusing primarily on the system architecture. We begin with a discussion of the issues concerning outdoor navigation, then describe the various perception, planning, and control components of our system that address these issues. We describe the CODGER software system for integrating these components into a single system, synchronizing the data flow between them in order to maximize parallelism. Our system is able to drive a robot vehicle continuously with two sensors, a color camera and a laser rangefinder, on a network of sidewalks, up a bicycle slope, and through a curved road through an area populated with trees. Finally, we discuss the results of our experiments, as well as problems uncovered in the process and our plans for addressing them.

1. Introduction

The goal of the Autonomous Land Vehicle group at Carnegie-Mellon University is to create an autonomous mobile robot system capable of operating in outdoor environments. Because of the complexity of real-world domains and the requirement for continuous and real-time motion, such a robot system needs system architectural support for multiple sensors and parallel processing. These capabilities are not found in simpler robot systems. At CMU, we are studying mobile robot system architecture and have developed the navigation system working at two test sites and on two experimental vehicles [2] [3] [4] [8] [10] [11]. This paper describes current status of our system and some problems uncovered through real experiments.

1.1. The Test Sites and Vehicles

We have two test sites, the Carnegie-Mellon University campus and an adjoining park, Schenley Park. The CMU campus test site has a sidewalk network including intersections, stairs and bicycle slopes (see Figure 1). The Schenley Park test site has curved sidewalks in an area well populated with trees (see Figure 2).

Figure 3 shows our two experimental vehicles, the NAVLAB used in the Schenley Park test site, and the Terregator used in the CMU campus test site. Both of them are equipped with a color TV camera and a laser rangefinder made by ERIM. The NAVLAB carries four general purpose computers (SUN-3s) on board. The Terregator is linked to SUN-3s in the laboratory with radio communication. All of the SUN-3s are interconnected with a EtherNet. Our navigation system works on both vehicles in each test site.

[1]This research was supported by the Strategic Computing Initiative of the Defense Advanced Research Project Agency, DoD, through ARPA Order 5351, and monitored by the U.S. Army Engineer Topographic Laboratories under contract DACA76-85-C-0003. Views and conclusions contained in this document are those of the authors and should not be interpreted as representing official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the United State Government.

1.2. Current System Capabilities

Currently, the system has the following capabilities.

- Able to execute a prespecified user mission over a mapped network of sidewalks, including turning at the intersections and driving up the bicycle slope.
- Able to recognize landmarks, stairs and intersections.
- Able to drive on unmapped, curved, ill-defined roads using assumptions about local road linearity.
- Able to detect obstacles and stop until they move away.
- Able to avoid obstacles.
- Able to drive continuously at 200mm/sec.

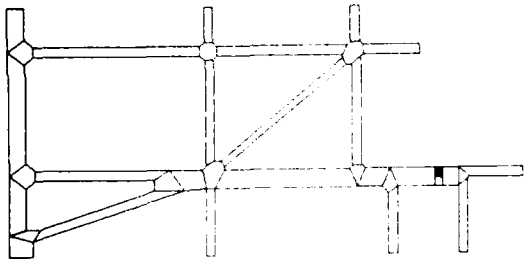


Figure 1: Map of the CMU Campus Test Site

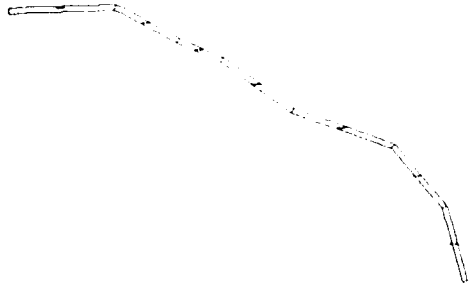
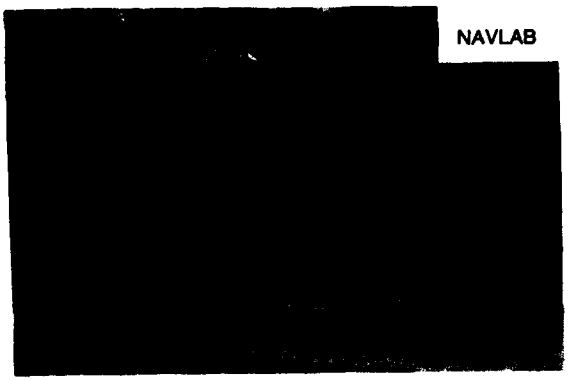


Figure 2: Map of the Schenley Park Test Site

2. Design of the System Architecture

In this section we describe the goals of our outdoor navigation system and the design principles, followed by an analysis of the outdoor navigation task itself. We describe our system architecture as it is shaped by these principles and analysis.



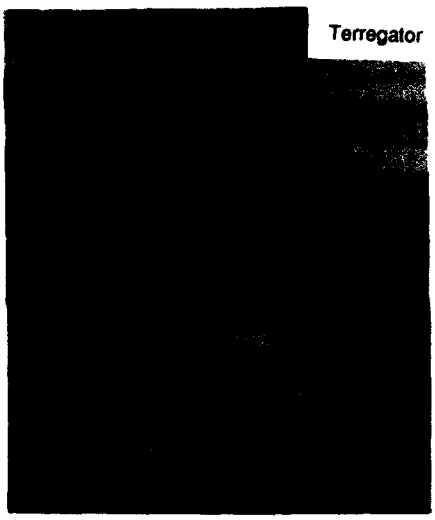


Figure 3: The NAVLAB and Terregator

2.1. Design Goals and Principles

The goals of our outdoor navigation system are:

- **map-driven mission execution:** The system drives the vehicle to reach a given goal position.
- **on- and off-road navigation:** Navigation environments include not only roads but also open terrain.
- **landmark recognition:** Landmark sightings are essential in order to correct for drift in the vehicle's dead-reckoning system.
- **obstacle avoidance**
- **continuous motion in real time:** Stop and go motion is unacceptable for our purposes. Perception, planning, and control should be carried out while the vehicle is moving at a reasonable speed.

In order to satisfy these goals, we have adopted the following design principles.

- **sensor fusion:** A single sensor is not enough to analyze complex outdoor environments. Sensors include not only a TV camera and a range sensor but also an inertial navigation sensor, a wheel rotation counter, etc.
- **parallel execution:** In order to process data from a number of sensors, make global and local plans , and drive the vehicle in real-time, parallelism is essential.
- **flexibility and extensibility:** This principle is essential because the whole system is quite large, requiring the integration of a wide range of modules.

2.2. Outdoor Navigation Tasks

Outdoor navigation includes several different navigation modes. Figure 4 illustrates several examples. On-road vs. off-road is just one example. Even in on-road navigation, *turning at the intersection* requires more sophisticated driving skill than *following the road*. In road following, the assumption that the ground is flat makes perception easier, but *driving through the forest* does not satisfy this assumption and requires more complex perception processing.

According to this analysis we decompose outdoor navigation into two navigation levels: *global* and *local*. At the global level, the system tasks are to select the best navigation route to reach the destination given by a user mission, and to divide whole route into a sequence of *route segments*, each corresponding to a uniform driving mode. The current system supports the following navigation modes: *following the road, turning at the intersection, driving up the slope.*

Local navigation involves driving within a single route segment.

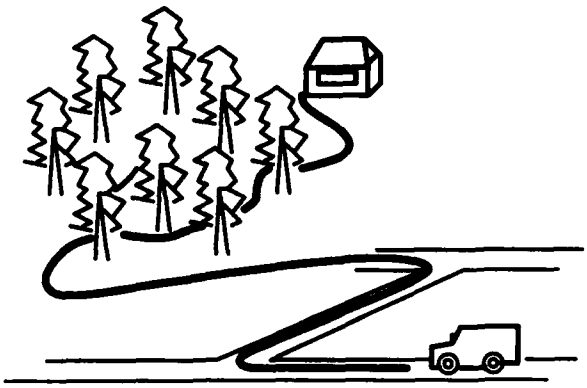


Figure 4: Outdoor Navigation

The navigation mode is uniform and the system drives the vehicle along the route segment continuously, perceiving objects, planning path plans, and controlling the vehicle. The important thing is that these tasks, perception, planning, and control, form a cycle and can be executed concurrently.

2.3. System Architecture

Figure 5 is a block diagram of our system architecture. The architecture consists of several modules and a communications database which links the modules together.

2.3.1. Module Structure

In order to support the tasks described in the previous section, we first decomposed the whole system into the following modules:

- **CAPTAIN** executes user mission commands and sends the destination and the constraints of each mission step to the MAP NAVIGATOR one step at a time, and gets the result of mission step.
- **MAP NAVIGATOR** selects the best route by searching the Map Database, decomposes it into a sequence of route segments, generates a route segment description which includes objects from the Map visible from the route segment, and sends it to the PILOT.

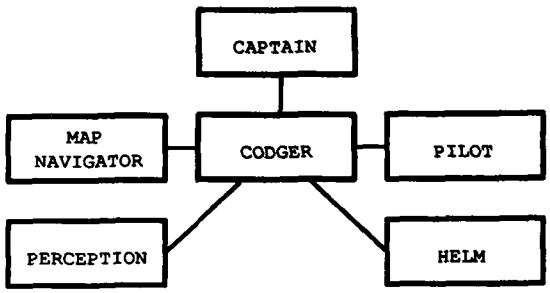


Figure 5: System Architecture

- **PILOT** coordinates the activities of PERCEPTION and the the HELM to perform local navigation continuously within a single route segment.
- **PERCEPTION** uses sensors to find objects predicted to lie within the vehicle's field of view. It estimates the vehicle's position if possible.
- **HELM** gets the local path plan generated by the PILOT and drives the vehicle.

The PILOT is decomposed into several submodules which run concurrently (see Figure 6).

- **DRIVING MONITOR** decomposes the route segment into small pieces called *driving units*. A driving unit is the basic unit for perception, planning, and control processing at the local navigation level. For example, PERCEPTION must be able to process a whole driving unit with a single image. The DRIVING MONITOR creates a *driving unit description* , which describes objects in the driving unit, and sends it to the following submodules.
- **DRIVING UNIT FINDER** functions as an interface to PERCEPTION, sending the driving unit description to it and getting the result from it.
- **POSITION ESTIMATOR** estimates the vehicle position using both of the result of PERCEPTION and dead-reckoning.
- **DRIVING UNIT NAVIGATOR** determines the admissible passage in which to drive the vehicle.
- **LOCAL PATH PLANNER** generates the path plan within the driving unit, avoids obstacles and keeps the vehicle in the admissible passage. The path plan is sent to the HELM.

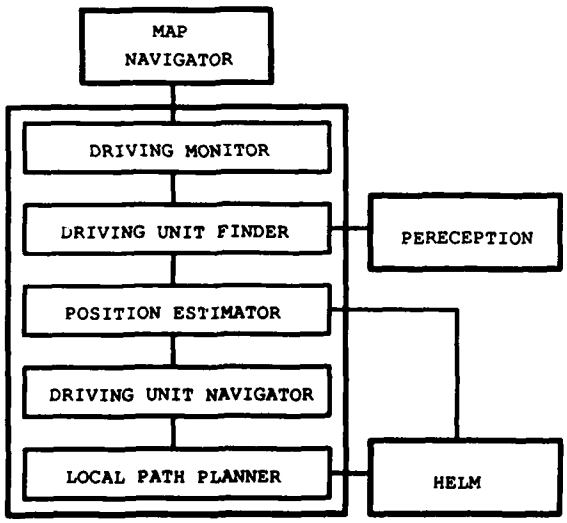


Figure 6: Submodule Structure of the PILOT

2.3.2. CODGER

The second problem in the system architecture design is connecting the modules. Based on our design principles, we have created a software system called *CODGER* (COmmunications Database with GEometric Reasoning) which supports parallel asynchronous execution and communication between the modules. We describe CODGER in detail in the next section.

3. Parallelism

3.1. The CODGER System for Parallel Processing

In order to navigate in real-time, we have employed parallelism in our perception, planning, and control subsystems. Our computing resources consist of several SUN-3 microcomputers, VAX minicomputers, and a high-speed, parallel processor known as the WARP interconnected with an EtherNet. We have designed and implemented a software system called CODGER (COmmunications Database with GEometric Reasoning) [9] to effectively utilize this parallelism.

The CODGER system consists of a central database (*Local Map*), a process that manages this database (*Local Map Builder or LMB*), and a library of functions for accessing the data (*LMB interface*) (see Figure 7). The various perceptual, planning, and control modules in the system are compiled with the LMB interface and invoke functions to store and retrieve data from the central database. The CODGER system can be run on any mix of SUN-3s and VAXes and handles data type conversions automatically. This system permits highly modular development requiring recompilation only for modules directly affected by a change.

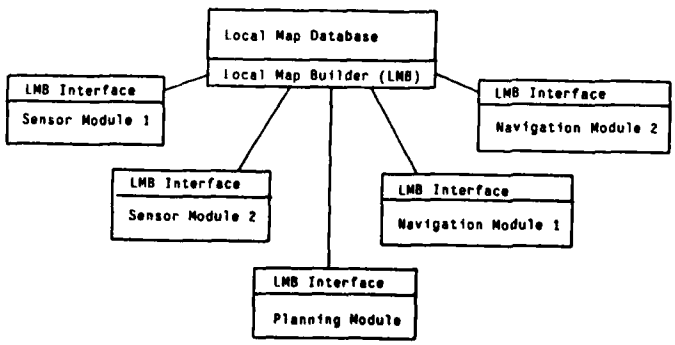


Figure 7: The CODGER Software System

3.1.1. Data Representation

Data in the Local Map is represented in *tokens* consisting of *lists of attribute-value* pairs. Tokens can be used to represent any information including physical objects, hypotheses, plans, commands, and reports. The token types are defined in a *template file* which is read by the LMB at system startup time. Attribute types may be the usual scalars (e.g., floats, integers), sets of scalars, or geometric locations. Geometric locations consist of a two- dimensional, polygonal *shape* and a reference coordinate *frame*. The CODGER system provides mechanisms for defining coordinate frames and for automatically converting geometric data from one frame to another, thereby allowing modules to retrieve data from the database and representing it in a form meaningful to them. Geometric data is the only data interpreted by the CODGER system; the interpretation of all other data types is delegated to the modules that use them.

3.1.2. Synchronization

The LMB interface provides functions for storing and retrieving data from the central database. Tokens can be retrieved using *specifications*. Specifications are simply boolean expressions evaluated across token attribute values. A specification may include computations such as mathematical expressions, boolean relations, and comparisons between attribute values. Geometric indexing is of particular importance for a mobile robot system. For example, the planner needs to search a database of map objects to locate suitable landmarks or to find the shortest path to the goal. The CODGER system provides a host of functions including those for computing the distance and intersection of locations. These functions can be embedded in specifications and matched to the database.

The CODGER system has a set of primitives to ensure that data transfer between system modules is synchronized and runs smoothly. The synchronization is implemented in the data retrieval mechanism. Specifications are sent to the LMB as either one-shot or standing requests. For one-shot specs, the calling module blocks while the LMB matches the spec to the tokens. Tokens that match are retrieved and the module resumes execution. If no tokens match, either the module stays blocked until a matching token appears in the database or an error is returned and the module resumes execution, depending on an option specified in the request. For example, the PATH PLANNER may use a one-shot to find obstacles stored in the database *before* it can plan a path. In contrast, the HELM, which controls the vehicle, uses a standing spec to retrieve tokens supplying steering commands *whenever* they appear.

3.2. Parallel Asynchronous Execution of Modules

Thus far we have run our scenarios with four SUN-3s interconnected with an EtherNet. The CAPTAIN, MAP NAVIGATOR, PILOT, and HELM are separate modules in the system, and PERCEPTION is two modules (range and camera image processing). All of the modules run in parallel; they synchronize themselves through the LMB database.

3.2.1. Global and Local Navigation

A good example of parallelism in the system is the interaction between the CAPTAIN, MAP NAVIGATOR, and PILOT. The CAPTAIN and MAP NAVIGATOR search the map database to plan a global path for the vehicle in accordance with the mission specification. The PILOT coordinates PERCEPTION, PATH PLANNING, and control through the HELM to navigate locally. The global and local navigation operations run in parallel. The MAP NAVIGATOR monitors the progress of the PILOT to ensure that the PILOT's transition from one route segment to the next occurs smoothly.

3.2.2. Driving Pipeline

Another good example of parallelism is within the PILOT itself. As described earlier, the PILOT monitors local navigation. For each driving unit, the PILOT performs four operations in the following order: predict it, recognize with the camera and scan it for obstacles with the rangefinder, establish driving constraints and plan a path through it, and oversee the vehicle's execution of it. In the PILOT, these four operations are separate modules linked together in a pipeline. While in steady state, the PILOT is predicting a driving unit 12 to 16 meters in front of the vehicle, recognizing a driving unit and scanning it for obstacles (in parallel) 8 to 12 meters in front, planning a path 4 to 8 meters in front, and driving to a point 4 meters in front. The stages of the pipeline synchronize themselves through the CODGER database.

The processing times for each stage vary as a function of the navigation task. In navigation on uncluttered roads, the vision subsystem requires about 10 seconds of real-time per image, the range subsystem requires about 6 seconds, and the local path planner requires less than a second. In this case, the stage time of the pipeline is that of the vision subsystem: 10 seconds. In cluttered environments, the local path planner may require 10 to 20 seconds or more, thereby becoming the bottleneck. In either case, the vehicle is not permitted to drive on to a driving unit unit it has propagated through all stages of the pipeline (i.e., all operations have been performed on it). For example, when driving around the corner of a building, the vision stage must wait until the vehicle reaches the corner in order to see the next driving unit. Once the vehicle reaches the corner, it must stop while waiting for the vision, scanning, and planning stages to process the driving unit before driving again.

4. Sensor Fusion

4.1. Types of Sensor Fusion

The NAVLAB and Terregator vehicles are equipped with a host of sensors including color cameras, a laser rangefinder, and motion sensors such as a gyro and shaft-encoder counter. In order to obtain a single, consistent interpretation of the vehicle's environment, the results of these sensors must be fused. We have identified three types of sensor fusion [8]:

- **Competitive:** Sensors provide data that either agrees or conflicts. This case arises when sensors provide data of the same modality. In the CMU systems, the task of determining the vehicle's position best characterizes this type of fusion. Readings from the vehicle's dead-reckoning system as well as landmark sightings provide estimates of the vehicle's position.
- **Complementary:** Sensors provide data of different modalities. The task of recognizing three-dimensional objects illustrates this kind of fusion. In the CMU systems, a set of stairs is recognized using a color camera and laser rangefinder. The color camera provides *image* information (e.g., color and texture) while the laser rangefinder provides *three-dimensional* information.
- **Independent:** A single sensor is used for each task. An example of a task requiring a single sensor is distant landmark recognition. In this case, only the camera is used for landmarks beyond the range of the laser rangefinder.

4.2. Examples of Sensor Fusion Tasks

4.2.1. Vehicle Position Estimation

In our road following scenarios, vehicle position estimation has been the most important sensor fusion task. By vehicle position, we mean the position and orientation of the vehicle in the ground plane (3 degrees of freedom) relative to the world coordinate frame. In the current system, there are two sources of position information. First, dead-reckoning provides vehicle-based position information. The CODGER system maintains a history of the steering commands issued to the vehicle, effectively recording the trajectory of the vehicle from its starting point.

Second, landmark sightings directly pinpoint the position of the vehicle with respect to the world at a point in time. In the campus test site, the system has access to a complete topographical map of the sidewalks and intersections on which it drives. The system uses a

color camera to sight the intersections and sidewalks and uses these sightings to correct the estimate of the vehicle's position. The intersections are of rank three, meaning that the position and orientation of the vehicle with respect to the intersection can be determined fully (to three degrees of freedom) from the sighting. Our tests have shown that such landmark sightings are far more accurate but less reliable than the current dead-reckoning system, that is, landmark sightings provide more accurate vehicle position estimates; however, the sightings occasionally fail. If the vehicle position estimates from the sighting and dead-reckoning disagree drastically, the *conflict* is settled in favor of the dead-reckoning system; otherwise, the result from the landmark sighting is used. In this case, the CODGER system adjusts its record of the vehicle's trajectory so that it agrees with the most recent landmark sighting, and discards all previous sightings.

The CODGER system is able to handle landmark sightings of rank less than three. The most common "landmark" in our scenarios is the sidewalk on which the vehicle drives. Since a sidewalk sighting provides only the orientation and perpendicular distance of the vehicle with respect to the sidewalk, the correction is of rank two. Therefore, the position of the vehicle is constrained to lie on a straight line. The CODGER system projects the position of the vehicle from dead-reckoning onto this line and uses the projected point as a full (rank three) correction. Since most of the error in the vehicle's motion is lateral drift from the road, this approximation works well.

4.2.2. Pilot Control

Complementary fusion is grounded in the Pilot's control functions. The Pilot ensures that the vehicle travels only where it is permitted and where it can. For example, the color camera is used to segment road from nonroad surfaces. The laser rangefinder scans the area in front of the vehicle for obstacles or unnavigable (i.e., rough or steep) terrain. The road surface is fused with the free space and is passed to the local path planner. Since the two sensor operations do not necessarily occur at the same time, the vehicle's dead-reckoning system also comes into play.

4.2.3. Colored Range Image

Another example of complementary fusion of camera and range data is the colored range image. A colored range image is created by "painting" a color image onto the depth map of a range image. The resultant image is used in our systems to recognize complicated three dimensional objects such as a set of stairs. In order to avoid the relatively large error in the vehicle's dead-reckoning system, the vehicle remains motionless while digitizing a corresponding pair of camera and range images [2].

4.3. Problems and Future Work

We have plans for improving our sensor fusion mechanisms. Currently, the CODGER system handles competing sensor data by retaining the most recent measurement and discarding all others. This is undesirable for the following reasons. First, a single bad measurement (e.g., landmark sighting) can easily throw the vehicle off track. Second, measurements can reinforce each other. By discarding old measurements, useful information is lost. A weighting scheme is needed for combining competing sensor data. In many cases, it is useful to model error in sensor data as gaussian noise. For example, error in dead-reckoning may arise from random error in the wheel velocities. Likewise quantization error in range and camera images can be modeled as gaussian noise. A number of schemes exist for fusing such data ranging from simple Kalman filtering techniques to full-blown Bayesian observation networks [1] [7].

5. Local Control

In this section we discuss some of the control problems in local navigation.

5.1. Adaptive Driving Units and Sensor View Frames

Management of driving units and sensor view frames is essential in local control. As described in section 2, the driving unit is a minimum control unit, a unit to perceive objects, generate a path plan, and drive the vehicle. The PERCEPTION module digitizes an image in each driving unit, and the vehicle's position is estimated and its trajectory is planned once in each driving unit. Therefore, an appropriate driving unit size is essential for stable control. For example, the sensor view frame cannot cover a very large driving unit. Conversely, small driving units place rigid constraints on the LOCAL PATH PLANNER, because of the short distance between the starting point and the goal point. The aiming of the sensor view frame determines the point at which to digitize an image and to update the vehicle position and path plan.

In the current system, the sensor view frame is always fixed with respect to the vehicle. The size of the driving unit is fixed for driving on roads (4~6 meters length), and is changed for turning at intersections so that the entire intersection can be see in a single image and to increase driving stability (see Figure 8). This method works well in almost all situations in current test site.

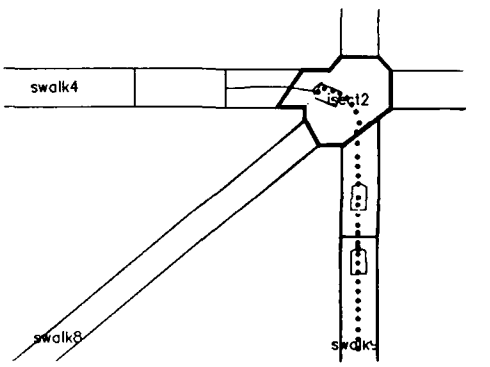


Figure 8: Intersection Driving Unit

For intersections requiring sharp turns (about 135 degrees), the current method does not suffice. Because there is only one driving *unit at the intersection*, the system digitizes an image, estimates the vehicle's position, and generates a path plan only once for a large turn. Furthermore, since the camera's field of view is fixed straight ahead, the system cannot see the driving unit after the intersection until the vehicle has turned through the intersection. Though actual path generated is not so bad, it is potentially unstable.

This experimental result indicates that the system should scan for an admissible passage, and update vehicle position estimation and local path plan more frequently when the vehicle changes its course faster. We plan to improve our method for managing driving units. Our new idea is:

- **Length of the driving unit:** The length of the driving unit is bounded at the low end by the LOCAL PATH PLANNER's requirements for generating a reasonable path plan, and at the high end by the view frame required by PERCEPTION for recognizing a given object.
- **driving unit interval:** The *driving unit interval* is the distance between the centers of adjacent driving units. Adjacent driving units can be overlapped, that is, they can be placed such that their interval is shorter than their length. Figure 9 illustrates this situation.
- **adjusting size and interval of driving unit:** If the passage is simple, the length and interval of the driving unit is long. If the passage is complex, for example, in the case of highly curved roads or intersections, or in the presence of obstacles, the length and interval of driving unit are shorter. And if the required driving unit interval must be shorter than the length of driving unit, the driving units are overlapped. Therefore, the vehicle's position is estimated and a local path is planned more frequently so that the vehicle drives stably (see Figure 9).

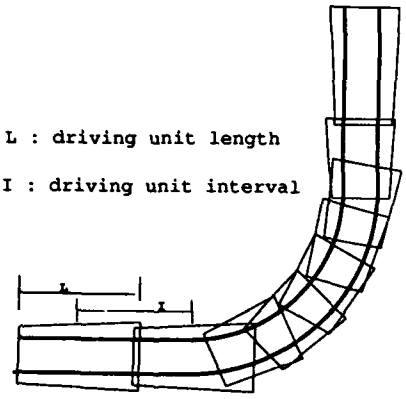


Figure 9: Adaptive Driving Units

- **adjusting sensor view frame:** The sensor view frame with respect to the vehicle, the distance and the direction to the driving unit from the vehicle, is adjusted using the *pan and tilt* mechanism of the sensor. In most cases, a longer distance to the next driving unit allows a higher vehicle speed. If the processing time of the PERCEPTION and the PILOT is constant, the longer distance means a higher vehicle speed. But the longer distance produces less accuracy in perception and vehicle position estimation. Therefore, the distance is determined for the required accuracy, which depends on the complexity of passage. Using the pan and tilt mechanism, PERCEPTION can digitize an image at the best distance from the driving unit, since the sensor's view frame is less rigidly tied to the orientation and position of the vehicle.

5.2. Vehicle Speed

It is an important capability for autonomous mobile robot to adjust the vehicle's speed automatically so that the vehicle drives safely at the highest possible speed. The current system slows the vehicle down in turning to reduce driving error.

The delay in processing in the LOCAL PATH PLANNER and communication between the HELM and actual vehicle mechanism *gives rise to error in vehicle position estimation. For example,* because of continuous motion and non-zero processing time, the vehicle position used by the LOCAL PATH PLANNER as a starting point differs slightly from the vehicle position when the vehicle starts executing the plan. Because the smaller turning radii give rise to larger errors in the vehicle's heading, which are more serious than displacement errors, the HELM slows vehicle for the smaller turning radii. This method is useful for making the vehicle motion stable.

We are going to add the capability to the system for adjusting the vehicle speed to the highest possible value automatically. Our idea is the following:

- **schedule token:** The modules and the submodules working at the local navigation level store their predicted processing times in a *schedule token* in each cycle. PERCEPTION is the most time consuming module, and its processing time varies drasticly from task to task.
- **adjusting vehicle speed:** Using the path plan and the predicted processing time stored in the schedule token, the HELM calculates and adjusts vehicle speed so that the speed is maximum and the modules can finish processing the driving unit before the vehicle reaches the *end of the current planned trajectory.*

5.3. Local Path Planning and Obstacle Avoidance

Local path planning is the task of finding a trajectory for the vehicle through admissible space to a goal point. In our system, the vehicle is constrained to move in the ground plane around obstacles (represented by polygons) while remaining within the driving unit (also a polygon). We have employed a configuration space approach [5] [6]. This algorithm, however, assumes that the vehicle is omnidirectional. Since our vehicles are not, we smooth the resultant path to ensure that the vehicle can execute it. The smoothed path is not guaranteed to miss obstacles. We plan to overcome this problem by developing a path planner that reasons about constraints on the vehicle's motion.

6. Navigation Map

Some information about vehicle's environment must be supplied to the system a priori, even if it is incomplete, and even if it is nothing more than a data format for storing explored terrain. The user mission, for example, "turn at the second cross intersection and stop in front of the three oak trees" does not make sense to the system without a description of environment. *The Navigation Map* is a data base to store the environment description needed for navigation.

6.1. Map Structure

The navigation map is a set of descriptions of physical objects in navigation world. It is composed of two parts, the geographical map and the object data base. The geographical map stores object locations with their contour polylines. The object data base stores object geometrical shapes and other attributes, for example, the navigation cost of objects. Though, in the current system, all objects are described with both of the geographical map and the object data base, in general, either of them can be unused. For example, the location of *stairs A* is known, but its shape is unknown.

The shape description is composed of two layers. The first layer stores shape attributes. For example, the width of the road, the length of the road, the height of the stairs , the number of steps, etc. The second layer stores actual geometrical shapes represented by the surface description. It is easy to describe incomplete shape information with only the first layer.

6.2. Data retrieval

The map data is stored in the CODGER data base as a set of tokens forming tree structure. In order to retrieve map data, parents tokens have indexes to children tokens. Because current CODGER system provides modules with a token retrieval mechanism that can pick up only one token at a time, retrieving large portions of the map is cumbersome. We plan to extend CODGER so that it can match and retrieve larger structures, possibly combined with an inheritance mechanism.

7. Other Tasks of the System

Navigation is just one goal of a *mobile robot system.* Generally speaking, however, navigation itself is not an end, but actually a means to achieve the final goals of the autonomous mobile robot system, such as carrying baggage, exploration, or refueling. Therefore, the system architecture must be able to accommodate tasks other than navigation.

Figure 10 illustrates one example of an extended system architecture which loads, carries and unloads baggage. The whole system is comprised of four layers, *mission control, vehicle resource management, signal processing,* and *physical hardware.* The CAPTAIN, only one module in the mission control layer, stores the user mission steps, sends them to the vehicle resource management layer one by one, and oversees their execution.

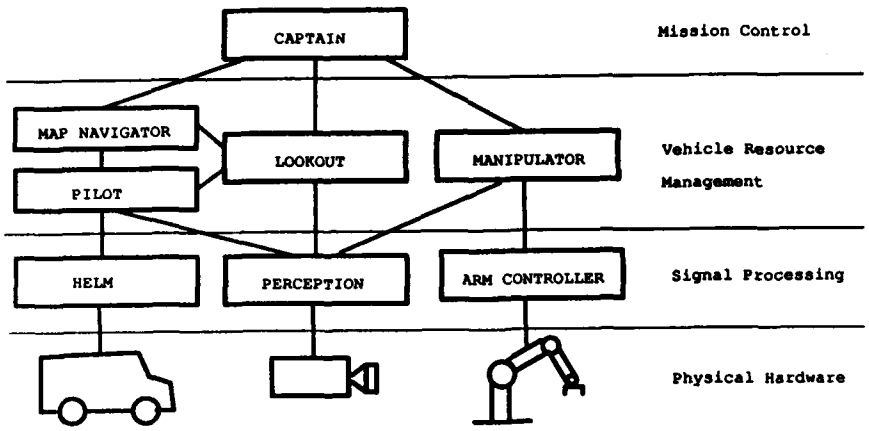


Figure 10: Extended System Architecture

In the vehicle resource management layer, there are different modules working for different tasks. Although their tasks are different, they all work in a symbolic domain and do not handle the physical world directly. These modules oversee mission execution, generate plans, and pass information to modules in the signal processing layer. Through CODGER, they can communicate with each other, if necessary. The MAP NAVIGATOR and the PILOT, parts of the navigation system, are included in the vehicle resource management layer. The MANIPULATOR makes a plan (e.g., how to load and unload baggage with the arm) and sends it to the ARM CONTROLLER.

The modules in the signal processing layer interact with physical world using senors and actuators. For example, PERCEPTION processes the signals from sensors, the HELM drives the physical vehicle, and the ARM CONTROLLER operates the robot arm. The bottom level contains the real hardware, even if it includes some primitive controller. The sensors, the physical vehicle, and the robot arm are included in this layer.

Because our current system architecture is built on the CODGER system it will be easy to expand it to include these additional capabilities.

8. Conclusions

In this paper, we have described the CMU architecture for autonomous outdoor navigation. The system is highly modular and includes components for both global and local navigation. Global navigation is carried out by a route planner that searches a map database to find the best path satisfying a mission and oversees its execution. Local navigation is carried out by modules that use a color camera and a laser rangefinder to recognize roads and landmarks, scan for obstacles, reason about geometry to plan paths, and oversee the vehicle's execution of a planned trajectory.

The perception, planning, and control components are integrated into a single system through the CODGER software system. CODGER provides a common data representation scheme for all modules in the system with special attention paid to geometry. CODGER also provides primitives for synchronizing the modules in a way that maximizes parallelism at both the local and global levels.

We have demonstrated our system's ability to drive around a network of sidewalks and along a curved road, recognize complicated landmarks, and avoid obstacles. Future work will focus on improving CODGER for handling more difficult sensor fusion problems. We will also work on better schemes for local navigation and will strive to reduce our dependence on map data.

9. Acknowledgements

The design of our architecture was shaped by contributions from the entire Autonomous Land Vehicle group at CMU. We extend special thanks to Steve Shafer, Chuck Thorpe, and Takeo Kanade.

I. References

[1] Durrant-Whyte, H.
Integration, Coordination and Control of Mult-Sensor Robot Systems.
PhD thesis, University of Pennsylvania, 1986.

[2] Goto, Y., Matsuzaki, K., Kweon, I., Obatake, T.
CMU Sidewalk Navigation System.
In *FJCC-86*. 1986.

[3] Hebert, M. and Kanade, T.
Outdoor Scene Analysis Using Range Data.
In *Proc. 1986 IEEE Conference on Robotics and Automation*. April, 1986.

[4] Kanade, T., Thorpe, C., and Whittaker, W.
Autonomous Land Vehicle Project at CMU.
In *Proc. 1986 ACM Computer Conference*. Cincinnati, February, 1986.

[5] Lozano-Perez, T., Wesley, M. A.
An Algorithm for Planning Collison-Free Paths Among Polyhedral Obstacles.
Communications of the ACM 22(10), October, 1979.

[6] Lozano-Perez, T.
Spatial Planning: A Configuration Space Approach.
IEEE Transactions on Computers C-32(2), February, 1983.

[7] Mikhail, E. M., Ackerman, F.
Observations and Least Squares.
University Press of America, 1976.

[8] Shafer, S., Stentz, A., Thorpe, C.
An Architecture for Sensor Fusion in a Mobile Robot.
In *Proc. IEEE International Conference on Robotics and Automation*. April, 1986.

[9] Stentz, A., Shafer, S.
Module Programmer's Guide to Local Map Builder for NAVLAB.
1986.
In Preparation.

[10] Wallace, R., Stentz, A., Thorpe, C., Moravec, H., Whittaker, W., Kanade, T.
First Results in Robot Road-Following.
In *Proc. IJCAI-85*. August, 1985.

[11] Wallace, R., Matsuzaki, K., Goto, Y., Webb, J., Crisman, J., Kanade, T.
Progress in Robot Road Following.
In *Proc. IEEE International Conference on Robotics and Automation*. April, 1986.

QUALITATIVE NAVIGATION

Tod S. Levitt, Daryl T. Lawton, David M. Chelberg, Philip C. Nelson

Advanced Decision Systems, Mountain View, California 94040

ABSTRACT

Existing robot navigation techniques such as triangulation, dead reckoning, ranging sensors, or stereo, depend on accurate range data, correspondence of map data with the robot's location, good estimates of the robot's motion, inertial navigation systems, or upon local obstacle avoidance techniques. These approaches tend to be brittle, accumulate error and utilize little or no perceptual information.

This paper describes a formal theory that depends on visual landmark recognition for representation of environmental locations, and encodes local perceptual knowledge in structures called viewframes. Paths in the world are represented as sequences of sets of landmarks, viewframes, and other distinctive visual events. Approximate headings are computed between viewframes that have lines of sight to common landmarks. Range-free, topological descriptions of place, called orientation regions, are rigorously abstracted from viewframes. They yield a coordinate-free model of visual landmark memory that can also be used for navigation and guidance. With this approach, a robot can opportunistically observe and execute visually cued "shortcuts". Map and metric data are not required, but, if available, are handled in a uniform representation within the qualitative navigation techniques.

1. INTRODUCTION

The questions that define navigation and guidance are:

- Where am I?
- Where are other places relative to me?
- How do I get to other places from here?

A robot that moves about the world must be able to compute answers to these questions. This paper is concerned with the structure and processing for robotic visual memory and inference that yields navigation and guidance. Here, guidance means the problem of visual path-planning. That is, the input data is assumed to be percepts extracted from imagery, and a database, i.e., memory, of models for visual recognition. A priori model and map data is only relevant insofar as it provides a basis for runtime recognition of observable events. This is distinguished from path traversability planning where the guidance questions concern computing shortest distances between points under constraints of support of the ground or surrounding environment for the robotic vehicle.

A *multi-level theory of spatial representation* based upon the observation and re-acquisition of distinctive visual events (i.e., landmarks) has been developed. The representation provides the theoretical foundations for a visual memory database that includes coordinate free, topological representation of relative spatial location, yet smoothly integrates available metric knowledge of relative or absolute angles and distances. Rules and algorithms are presented that, under the assumption of correct association of landmarks on re-acquisition (although not assuming landmarks are necessarily re-acquired) provide a robot with navigation and guidance capability. The visual memory constructed while moving through the environment contains sufficient data to deduce or update a map of the environment.

Existing robot navigation techniques include triangulation Matthies and Shafer - 86, ranging sensors Hebert and Kanade - 86, auto-focus Pentland - 85, stereo techniques Lucas and Kanade - 84, Eastman and Waxman - 85, dead reckoning, inertial navigation, geo-satellite location, correspondence of map data with the robot's location, and local obstacle avoidance techniques Moravec - 80. These approaches tend to be brittle Bajcsy et.al. - 86, accumulate error Smith and Cheeseman - 85, are limited by the range of an active sensor, depend on accurate measurement of distance/direction perceived or traveled, and are non-perceptual, or only utilize very weak perceptual models.

Furthermore, these theories are largely concerned with the problem of measurement and do not centrally address issues of map or visual memory and the use of this memory for inference in vision-based navigation and guidance. Exceptions to this are the work of Davis - 86, McDermott and Davis - 84, and Kuipers - 77. Davis addressed the problem of representation and assimilation of 2D geometric memory, but assumed an orthographic view of the world and did not consider navigation or guidance. McDermott and Davis developed an ad hoc mixture of vector and topological based route planning, but assumed a map, rather than vision derived world (in their assumptions of knowledge of boundaries, their shapes, and spatial relationships), had no formal theory relating the multiple levels of representation, and consequently did not derive or

implement results about path execution. Kuipers developed qualitative techniques for navigation and guidance. He assumed capability of landmark recognition, as we do, but relied on dead-reckoning and constraint to one-dimensional (road) networks to permit path planning and execution.

In this paper, we develop representation and inference for relative geographic position information that:

- build a memory of the environment the robot passes through,
- contains sufficient information to allow the robot to re-trace its paths,
- can be used to construct or update an a posteriori map of the geographic area the robot has passed through, and
- can utilize all available information, including that from runtime perceptual inferences and a priori map data, to perform navigation and guidance.

To accomplish this, the notion of a geographic "place" is defined in terms of data about visible landmarks. A place, as a point on the surface of the ground, is defined by the landmarks and spatial relationships between landmarks that can be observed from a fixed location. More generally we can define a place as a region in space, in which a fixed set of landmarks can be observed from anywhere in the region, and relationships between them do not change in some appropriate qualitative sense. Data about places is stored in structures called viewframes, boundaries and orientation regions.

Viewframes provide a definition of place in terms of relative angles and angular error between landmarks, and very coarse estimates of the absolute range of the landmarks from our point of observation. Boundaries and orientation regions provide a more qualitative definition of place. Both concepts allow us to localize ourselves in space relative to a set of observed landmarks, without necessarily using a priori map data.

Viewframes allow us to localize our position in space relative to observable local landmark coordinate systems. In performing a viewframe localization, we can make use of observed or inferred data about our approximate range to landmarks. Errors in ranging and relative angular separation between landmarks are smoothly accounted for. A priori map data can also be incorporated.

If we drop all range information, we can still use the notion of boundaries to determine our qualitative position relative to other landmarks. A pair of landmarks creates a virtual division of the ground surface by the line connecting the two landmarks. The observable relative orientation of the landmarks, i.e., the left-to-right order of the pair of landmarks, indicates which side of the landmark-pair-boundary (LPB) we are on. A set of LPB's with orientations determines a region on the ground called an orientation region. LPB's can be derived by considering pairs of landmarks from viewframes; in this case we speak of the set of orientation regions induced by or associated to the viewframe.

As a robot moves over the ground surface, the crossing of LPB's indicates passage from one orientation region to another. Thus, orientation regions are bounded by the unique observable visual events of passing to the right of, left of, or between a pair of landmarks. In this manner, LPB's and orientation regions yield a natural notion of headings and paths in the environment.

Paths in the world are represented as sequences of observations of landmarks, viewframes, LPB's and other distinctive visual events. We compute approximate headings between viewframes that have lines of sight to common landmarks. This allows a robot to opportunistically observe visually cued "shortcuts".

We define headings as world states that can be created by a robotic action, namely that of following the heading direction specifier, and whose negation, i.e., failure to maintain the heading specifier while in motion, can either be created by robotic actions or perceptually observed in environmental events. We make computational the concept of maintaining a heading until an observable or inferable termination condition is met. While a heading is being executed, the vision system builds up a visual representation of the environ-

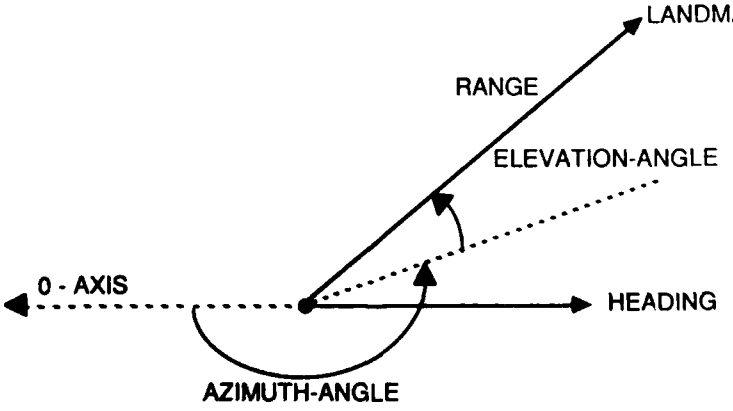


Figure 2-1: Sensor Centered Coordinate System

ment it is passing through. Termination conditions correspond to observable or computable changes in our location in space, triggering additional spatial inference processes if we have not reached our destination goal.

The robust, qualitative properties and formal mathematical basis of this representation and inference processes are suggestive of the navigation and guidance behavior in animals and humans [Schone - 84]. However, we make no claims of biological foundations for this approach.

In the following, we first develop the mathematical theory of viewframes, boundaries and orientation regions. We then show how these qualitative, topological concepts interact with a priori metric map data. Perceptual definitions of landmarks and requirements for vision system performance in re-acquisition of landmarks to support the proposed perception-based navigation approach are reviewed in Section 3. Inference for path planning over viewframe, boundary and orientation region representations are presented in Section 4, and results are presented in Section 5.

2. TOPOLOGICAL LANDMARK NETWORK REPRESENTATIONS

2.1 VIEWFRAMES

A viewframe encodes the observable landmark information in a stationary panorama. That is, we assume that the sensor platform is stationary long enough for the sensor to pan up to 360 degrees, to tilt up to 90 degrees (or to use an omni-directional sensor [Cao et.al. - 86]), to recognize landmarks in its field of view, or to buffer imagery and recognize landmarks while in motion.

To generate a viewframe, relative solid angles between distinguished points on landmarks are computed using a sensor-centered coordinate system. A distinguished point on a landmark may be a statistically derived point such as the centroid of the projection of the landmark in an image, a structurally derived point such as a vertex or high curvature boundary point, or perceptually derived as in a unique visual feature of the landmark. Representation of landmarks and choice of distinguished points is presented in greater detail in Section 3.

A sensor-centered spherical coordinate system is established. It fixes an orientation in azimuth and elevation, and takes the direction opposite the current heading as the zero degree axis. Then two landmarks in front of us, relative to our heading, will have an azimuth separation of less than 180 degrees. If we assume that no two distinguished landmark points have the same elevation coordinates (i.e., no two distinguished points appear one directly above the other) then we obtain a well-ordering of the landmarks in the azimuth direction, which we can speak of as "ordered from left to right". The relative solid angle between two distinguished landmark points is now well defined; see Figure 2-1.

Under the above assumptions, we can pan from left to right, recognizing landmarks, L_i, and storing the solid angles between landmarks in order, denoting the angle between the i-th and j-th landmarks by Ang_{ij}. The basic viewframe data are these two ordered lists, $(L_1, L_2, \ldots)$ and $(Ang_{12}, Ang_{23}, \ldots)$. The relative angular displacement between any two landmarks can be computed from this basic list. However, the angle between landmarks that is more readily measured is the planar angle, θ_{ij}, established by the sensor focal point and the two distinguished points observed in the i-th and j-th landmarks. There is an error in computing these relative angles that is at least as great as resolution of the vision system, and may include cumulative pan/tilt error, angular ambiguity in landmark point localization, or other error sources. The angular error is measured by e_{ij} between landmarks i and j. Finally, range estimates are required for landmarks recorded in viewframes. These estimates can be arbitrarily coarse, but finite. We only require that the true range lie between the bounds specified for the estimate. We denote the range interval associated to landmark L_i by $[r_{i1}, r_{i2}]$. We now explain how it is possible to localize ourselves in space relative to these observed landmarks.

We begin by noting that the set of points in 3-space from which we can observe an angle of θ_{ij} between landmarks L_i and L_j is constrained to a closed torus-like surface; a cut-away of this surface is pictured in Figure 2-2. This is more easily observed in a planar cross-section, where the shape is the figure eight cross-section in Figure 2-3. To prove this, we set up a polar coordinate system with the origin at L_i, and with L_j at coordinates $[s_{ij}, 0]$, where s_{ij} is the fixed,

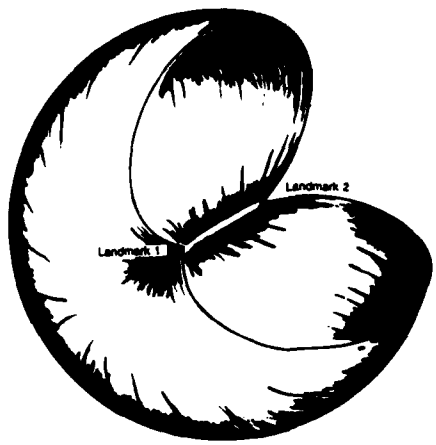


Figure 2-2: Constant Angle Toroid

but unknown, distance between the landmarks. Denote the (unknown) distance from the sensor to L_i by r_i and the distance to L_j by r_j. We now ask, what are the set of points in the plane, $[r, ang]$, from which we can observe an angle of θ_{ij} between L_i and L_j? This

situation is pictured in Figure 2-3. We can now compute:

$$r_i = s_{ij}\cos\phi + s_{ij}\cot\theta\sin\phi$$

$$\phi = \cot^{-1}\left[\frac{r_i}{r_j}\csc\theta - \cot\theta\right]$$

The first equation is the polar form for a circle with polar center $\left[\frac{s_{ij}}{2}\csc\theta, \frac{\pi}{2}-\theta\right]$ and radius $\frac{s_{ij}}{2}\csc\theta$. It has singularities where r_i or r_j are equal to zero. By symmetry we obtain the figure eight like shape pictured in Figure 2-3. Rotating the circular arc in 3-space about the axis defined by the line segment joining L_i and L_j, we obtain the figure pictured in Figure 2-2.

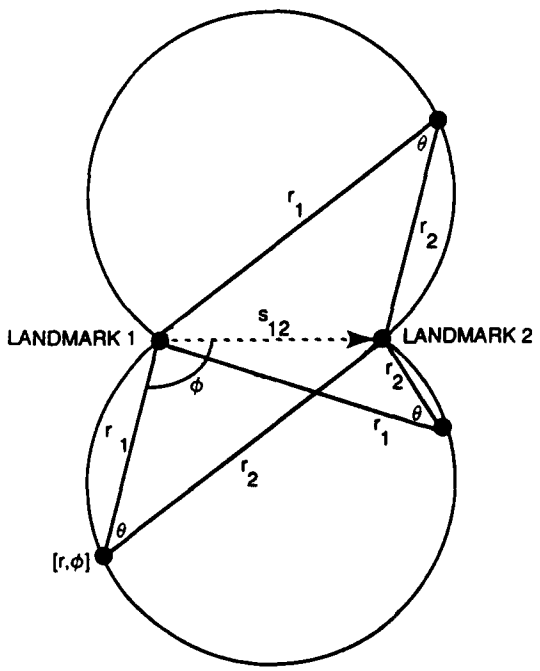


Figure 2-3: Constant Angle Circular Arcs

Figure 2-3 shows how varying the sensor location along the circular arc is equivalent to varying the absolute ranges to the two landmarks. If we can bound the ranges to the landmarks, then we can localize ourselves along the circular arc accordingly. This is logically equivalent to establishing a local coordinate frame between the landmarks, and bounding our location relative to that frame. If we can register the landmarks with a priori map data, then we can know and use the distance between the landmarks, but this is not necessary.

Error in angular measurement of the θ_{ij} corresponds to different choices of concentric circular arcs each of which contains L_i, L_j and the sensor focal point. If we union these arcs together, we obtain the localization, Loc_{ij}, of the sensor relative to the two landmarks. Because our localization must be true relative to all observed landmark pairs simultaneously, the intersection of the Loc_{ij} over all $i > j$, give our best localization. Such a localization is pictured in Figure 5-2. Intersecting the Loc_{ij} requires that the Loc_{ij} be represented in the same local coordinate frames. We discuss transformations between local landmark coordinate frames in Section 4. These concepts can clearly be extended to 3-space reasoning, if necessary.

2.2 BOUNDARIES AND ORIENTATION REGIONS

Viewframes contain two basic dimensions of data: the relative angles between landmarks, and the estimated range (intervals) to the landmarks. If we drop the range information, we are left with purely topological data. That is, it is impossible, using only the relative angles between landmarks, and no range, map or other metric data, to determine the relative angles between triples of landmarks, or to construct parametric representations of our location with respect to the landmarks. Nonetheless, there is topological localization information present in the ordinal sequence of landmarks; there is a sense in which we can compute differences between geographic regions, and observe which region we are in.

The basic concept is to note that if we draw a line between two (point) landmarks, and *project* that line onto the (possibly not flat) surface of the ground, then this line divides the earth into two distinct regions. If we can observe the landmarks, we can observe which side of this line we are on. The "virtual boundary" created by associating two observable landmarks together thus divides space over the region in which both landmarks are visible. We call these landmark-pair-boundaries (LPB's), and denote the LPB constructed from the landmarks L_1 and L_2 by LPB(L_1,L_2).

Roughly speaking, if we observe that landmark L_1 is on our left hand, and landmark L_2 is on our right, and the angle from L_1 to L_2 (left to right) is less than 180 degrees, then we denote this side of, or equivalently, this orientation of, the LPB by $[L_1\ L_2]$. If we stand on the other side of the boundary, LPB(L_1,L_2), "facing" the boundary, then L_2 will be on our left hand and L_1 on our right and the angle between them less than 180 degrees, and we can denote this orientation or side as $[L_2\ L_1]$ (left to right).

More rigorously, define:

orientation-of-LPB(L_1,L_2)

$$= \text{sign}(\pi - \Theta_{12}) = \begin{cases} +1 & \text{if } \Theta_{12} < \pi \\ 0 & \text{if } \Theta_{12} = \pi \\ -1 & \text{if } \Theta_{12} > \pi \end{cases}$$

where Θ_{12} is the relative azimuth angle between L_1 and L_2 measured in an arbitrary sensor-centered coordinate system. Here, an orientation of +1 corresponds to the $L_1 L_2$ side of LPB(L_1,L_2), -1 corresponds to the $L_2 L_1$ side of LPB(L_1,L_2) and 0 corresponds to being on LPB(L_1,L_2). It is a straightforward to show that this definition of LPB orientation does not depend on the choice of sensor-centered coordinate system.

LPB's give rise to a topological division of the ground surface into observable regions of localization, called orientation regions. Crossing boundaries between orientation regions leads to a qualitative sense of path planning based on perceptual information. Inference for perceptual path planning is explored in Section 4.2.

Figure 2-4(a) shows the LPB's that are implicit in a viewframe. The solid lines are the virtual boundaries created by landmark pairs. Figure 2-4(a) can be misleading in that it seems to imply that the LPB's contain the data to compute the angle-distance geometry of the sensor location relative to the landmarks. Figure 2-4(b) shows a representation of the same viewframe. Here the ranges to the landmarks have been changed, but the ordinal angular relationships between landmarks have not. The angle-distance geometry of our apparent location relative to the landmarks is completely different; however, the topological information, that is, the number of regions, their number of sides and adjacency relations, are preserved.

More specifically, there is topological information captured in the orientation region representation that distinguishs regions, yet is invariant under large motions of the observing sensor. The LPB's divide the ground surface into regions. If we regard the boundaries of regions as fattened wireframes, then the orientation regions may be thought of as (topological) holes in the surface. If we view the surface as being the whole earth, then the shape formed by cutting the orientation regions out of the surface of the (hollow) earth, is a two dimensional manifold with boundary; in particular, the shape is topologically equivalent to a (two) sphere with finitely many holes cut out of it. The number of holes is the number of orientation regions. From topology we have that two orientable two dimensional manifolds with boundary, are topologically equivalent if and only if they have the same genus and the same number of holes (i.e., boundary components) Massey - 67 . In this case, the genus of a sphere with holes in it is zero, so two shapes induced by orientation regions are topologically equivalent if and only if they have the same number of orientation regions.

Notice that orientation regions are equivalence classes of viewframes, where two viewframes are equivalent if the anzimuth angular order of visible landmarks is the same. There is a strong relationship between the number of orientation regions the LPB's divide space into, and the angular order, left to right in azimuth, observed among the landmarks in the viewframe.

To see this, begin with a single LPB. This clearly divides space into two regions. Reasoning inductively, suppose we have N LPB's already established. If we

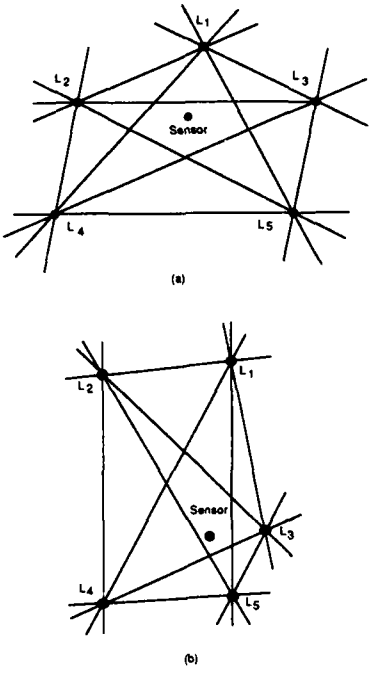


Figure 2-4: Viewframe Orientation Regions

introduce another LPB, each time it crosses an existing LPB, it divides the last region it was crossing through into two, thus generating exactly one more region. If it crosses through a boundary at a point that is already the crossing point for two or more other LPB's, then, although it is technically crossing two (or more) LPB's simultaneously, it generates only one more region. Finally note that after the LPB has no more other LPB's to cross (which will happen because it can only cross each LPB at most once), it continues on "to infinity", creating exactly one more (unbounded) region. Thus, the total number of regions created by N LPB's is equal to:

number-orientation-regions = N + (number-of-crossings-with-multiplicites) + 1

Here the multiplicity of a crossing is defined as 1, if two LPB's cross, and, in general, as (number-of-LPBs-crossing - 1) for two or more LPB's.

If a viewframe contains the landmark sequence L_1, L_2, L_3, L_4, then LPB(L_1, L_3) and LPB(L_2, L_4) must cross. It is possible for LPB(L_1, L_2) and LPB(L_3, L_4) to be parallel, and crossing points of greater than multiplicity 1 can happen; however, these are relatively rare events. It follows that a reasonable estimate of the number of orientation regions implicit in a viewframe can be gotten by assuming that all LPB crossings generated by pairs of landmarks in the viewframe have multiplicity 1 (except for the crossings through landmarks themselves, which have multiplicity K-1 for K landmarks), and that no LPB's are parallel. Under these assumptions, the number of orientation regions determined by a viewframe only depends on the number of landmarks in the viewframe.

Notice that crossings of multiplicity 1 are stable in that small perturbations of the locations of the four landmarks that generate the two crossing LPB's cannot change the multiplicity of the crossing. However, crossings of multiplicity greater than 1 are inherently singular (up to the vision system's ability to resolve landmark points). Furthermore, the crossing of two LPB's is also stable, while the event of two LPB's being parallel is a singular event. Interestingly, the parallelism of two LPB's is not observable without relative range information.

Now if we observe a viewframe and ignore the range information, we can still observe all $\binom{n}{2}$ LPB's created by taking landmarks in pairs. The conjunction of the orientations defines the boundaries of the orientation region we currently occupy. Any conjunction of LPB orientations that shares at least one LPB, but has the orientation reversed, must be in a different region of space.

We call the set of orientation regions created by a set of landmarks, the orientation-net associated to the set of landmarks. If we drop one of the LPB's in the set defining an orientation-net, we still have a localization of space defined by the remaining orientation regions. The previous localization must be a refinement of the current one. Thus, the partially ordered set of subsets of LPB's induce a partial order on refinements and coarsenings of the orientation region nets on the ground.

We can conclude from this analysis that if we are at a "stable viewpoint" relative to the set of landmarks recorded in a viewframe, then the angular order of the landmarks completely determines the orientation-region we are in, within the orientation-net determined by the landmarks in the viewframe. Thus we see that if we assume only stable crossings of LPB's then the multi-resolution (i.e., partially ordered) set of orientation-nets derived from a viewframe induces a unique topological representation of the visible space. A subset of the total observable set of LPB's can be used for spatial inference. This corresponds to moving to a coarser level of orientation-net in which several finer regions may become indistinguishable.

2.3 RELATIONSHIP TO METRIC REPRESENTATIONS

We define a metric representation to be one that contains enough data to compute exact (although not necessarily correct) geometric angle-distance relationships between the represented points. For example, map data is metric, environmental representations obtained from ranging sensors are metric, a list of UTM coordinates of bridge locations is metric, and an exact angle/distance graph of locally maximum elevation points extracted from a government terrain grid database is a metric representation.

There are numerous ways to integrate metric information with the more qualitative viewframe, boundary and orientation region representations to augment the ability of a land-based robot to navigate, to guide itself about the world, and to cue addition and refinement of map data with visually acquired knowledge. Here we present several methods that make use of the metric data while retaining the robustness of the more qualitative representations. These include using metric data for:

- refinement of range estimates in viewframes
- prediction of appearance and location of new (i.e., currently occluded or as yet unobserved) landmarks
- detection of conflict between predicted landmark locations and LPB crossings, and
- computing headings.

The first three are presented below. The last is explained in Section 4.

If we can correspond landmarks observed in a viewframe with topographic map or grid database locations, then we know the distance between the observed landmarks. Knowing the distance between landmarks allows to refine our estimates of range intervals to observed landmarks as follows. The constraint that the distance between the landmarks is a fixed distance, s_{12}, can be represented by the law of cosines applied to the triangle formed by the sensor focal point and the two landmarks, as in Figure 2-3. If r_1 is a range estimate to L_1 and r_2 is a range estimate to L_2, then we have

$$s_{12}^2 = r_1^2 - r_2^2 - 2r_1 r_2 \cos\theta_{12}$$

Let (r_{11}, r_{12}) be the range interval for L_1 and (r_{21}, r_{22}) be the range interval for landmark L_2. Then fixing r_1 at r_{11} and solving for r_2 yields a new lower bound for the range interval for L_2, and fixing r_1 at r_{12} yields a new upper bound for the interval. (If there are two possible values for r_2, take the minimum or maximum respectively.) If either of the new bounds are worse than the original estimates, then the information may be used to refine the bounds on the range interval for L_1 instead of L_2. However, if we order the landmark pairs from smallest to largest range intervals, then no back-tracking will be necessary, and an algorithm for refining the intervals in a viewframe can be done in linear time in the number of landmark pairs.

If constraints on viewframe ranges from metric data result in a "negative" interval (i.e., maximum range being less than the minimum) this indicates that either the true range value to a landmark does not lie in the estimated range interval or that the metric data is incorrect. This conflict can be used to cue alternative inference strategies in path planning, such as only relying on coherent data in the path planning.

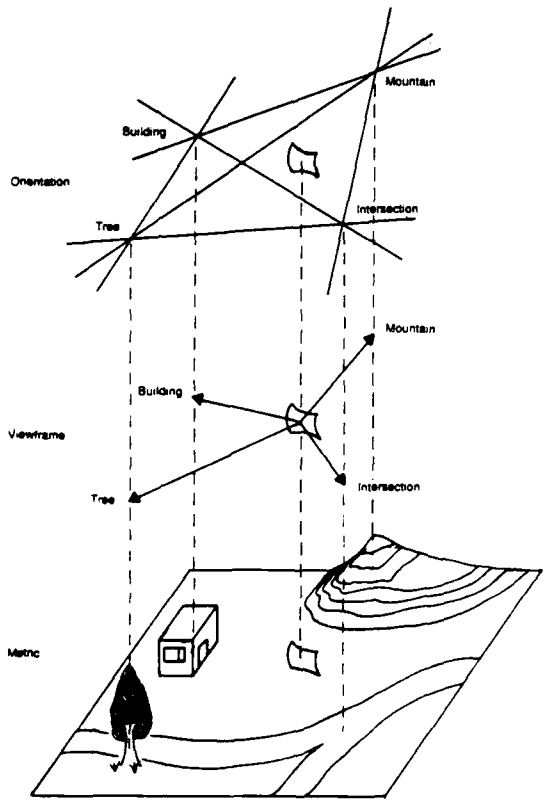


Figure 2-5: Multiple Levels of Spatial Representation

To use metric data for prediction of landmarks in imagery, we begin by mapping the viewframe-localization regions into a digitized terrain map. This is accomplished by computing the landmark pair circles (see Figure 2-3) and representing them as a bit plane overlay on the terrain grid. Because the landmarks have fixed locations on the grid, we can AND the bit planes corresponding to multiple landmark pairs, range intervals and angular errors, to find the set of grid points that corresponds to the viewframe-localization region. Computationally, we have saved the additional error we obtain by computing multiple local coordinate transforms between landmark pairs and intersecting the results algebraically. This is possible because mapping the viewframes into the metric data gives us an absolute coordinate system to work in.

Landmarks such as horizon extrema, change of groundcover, and manmade objects can be predicted a priori in the grid data. We can use the metric elevation data to compute the visibility of landmarks from selected points in the viewframe overlaid on the grid data. See [Lawton et.al. - 86], as an example of an implementation of such a visibility and image prediction. These predictions also yield range estimates that depend on the size of the viewframe-localization.

Finally, mapping viewframes into metric data allows us to predict, given a heading, the LPB's that we should cross first. We again represent the heading as a bit-plane overlay of a line (or other one-dimensional curve) on the terrain grid. LPB's between landmarks are also drawn as overlays. Following along the heading line orders the boundary crossings obtained by AND-ing the overlays. Because the viewframe-localization is, in general, a region rather than a point, headings can be drawn from multiple points in the region of localization. LPB crossings that agree in order across the multiple heading possibilities can be used as predictors to flag conflicts as potential errors in landmark recognition (i.e., bad association to the metric data), or as inaccuracies in the metric data.

The three spatial representation levels of metric data, viewframes and orientation regions are shown in Figure 2-5. In Section 4, we discuss inference strategies that use information at all three levels to perform navigation and guidance.

3. REQUIREMENTS FOR PERCEPTION

Ideal landmarks for navigation and map-building are uniquely distinguishable points that are visible from anywhere with precisely determined range. The real world, of course, consists of objects that occlude each other and look very different under varying viewing conditions. Significant perceptual and cognitive inference is required to recognize the same objects from different places and environmental conditions. For navigation and map building in particular, the changes in object appearance can be considerable because the location refining power of landmarks depends on relating landmarks from highly separated points of observation. In this section, we present some of the requirements and assumptions we use concerning perceptual processing for landmark extraction and recognition.

3.1 LANDMARKS

Landmarks are distinctive and stable objects or perceptual events. One distinguishing dimension for landmarks is the extent to which they are perceptual events or instances of known types of objects. Perceptual processing can extract landmarks using distinctive image events that are unique in multiple fields of view and are stable during observer motion but are not necessarily related to any type of modeled object. Landmarks that are an instance of an object model have default model-based information associated with them that can simplify and direct the accumulation of information as views of the landmark changes, especially due to model-based scale constraints. If a perceptually, but not model-based, landmark is not stable

from different views, it will be represented multiple times as different landmarks. Another perceptual processing requirement for landmarks in the topological representation is to be able to extract and monitor specific qualitative singularities, such as co-linearity of landmarks, or an LPB crossing.

A further classification for landmarks is whether they are of type point, composite, or extended (see Figure 3-1). Point landmarks are highly identifiable and localizable. Composite landmarks consist of tightly connected point landmarks corresponding to sub-parts, features or details. Composite properties are important for localization with respect to landmarks with orientational features such as the faces of a distant building. Extended landmarks correspond to rivers, roads, mountain ranges, and distinguished terrain-type boundaries. Knowledge of them is important for path planning because of the global effects they have on allowable paths.

Perceptually, a localization relative to extended landmarks involves adjacency, sidedness, and average distance estimates. Extended landmarks can be stable from image to image, but provide no basis for precise localization unless point landmark events are simultaneously observed. It is perceptually important to note the relations between these different types of landmarks over time, and to accumulate this information as part of the stored landmark. Thus, the set of point landmarks associated with composite and extended landmarks must be explicitly represented.

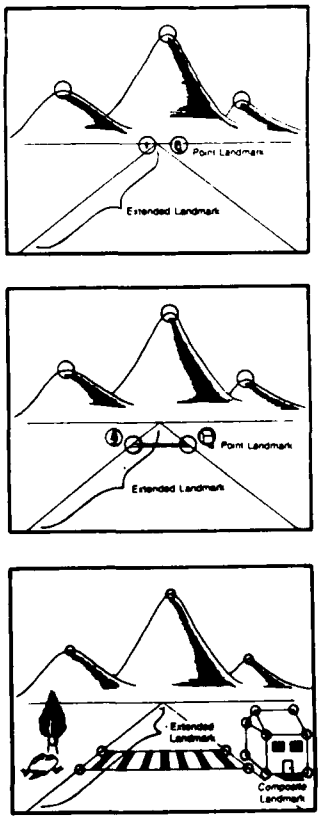


Figure 3-1: Landmark Types

3.2 PERCEPTION SYSTEM

Our perceptual system and its relation to landmark extraction is sketched in Figure 3-2. It consists of an organized hierarchy of perceptual events. The bottom level events are a time-indexed buffer of images and image registered features; the next level consists of several types of basic perceptual objects, such as points, curves and regions. These are processed into spatial temporal perceptual groupings that reflect

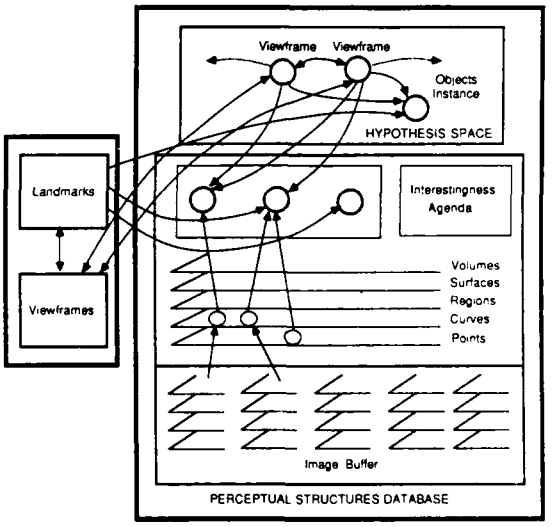


Figure 3-2: Perceptual Processing for Landmarks

stable perceptual structures in images over time. The perceptual objects and groups refer to positions in a sequence of images, but are independent of a particular view. Associated with perceptual objects is a mechanism for selecting those that are inherently interesting. This controls the application of grouping processes to limit the complexity of matching the precondtions of grouping rules to objects and relations. It also serves to extract significant perceptual structures to cue matching to object models.

The hypothesis space consists of instantiations of object models and viewframes that describe the relative positions of landmarks from a point of observation. A viewframe abstracts from the current scene model the details associated with landmarks. Viewframes are connected to each other along a path with pointers to landmark correspondence and changes between viewframes. The hypothesis space includes lists of the currently visible landmarks associated with the interestingness mechanism; this mechanism is used to note the changes in landmarks over time, such as occlusion or disocclusion and the emergence or disappearance of detail. The occurence of these events determines when new landmarks should be extracted or new viewframes established.

3.3 SIMPLIFYING ASSUMPTIONS

There are several assumptions about the types of sensors and their arrangement that simplify landmark determintation. If we assume the observer knows the relation between gravity and positions in an image, we can orient each pixel with respect to the direction of gravity to determine the vertical orientation of distant landmarks. We can also constrain the localization of the global horizon line and the sky. We have found that filtering out the set of the n-most interesting structures above and near the global horizon line that are distinct from sky, provides a useful set of landmarks. In addition, the sensor's orientation to the immediate ground-plane can be used to define a local horizon, thus determining the position of more local landmarks.

It is helpful for the visual sensors to have as complete a view of the world as possible. A spherical imaging surface provides an optimal arrangement [Cao et.al. - 86]. It can be approximated by a complete coverage of the field of view using multiple calibrated cameras. This resolves many problems in piecing together multiple local views to infer an omni-complete view of the environment. The sensors should also be stabilized relative to the environment so observer motions are effectively a sequence of, potentially different, translations. With a complete coverage of the field of view, there is a lessened need to reorientate the sensors.

Monitoring the displacement of image features relative to a known axis of translation simplifies frame to frame matching, Bolles and Baker - 85] [Lawton - 83], and provides basic information for determining how distant a landmark is, for how long a time it is visible, and when potential occlusions may occur. The classes of image transformations are contrained in these cases. Further, for the spatial representation we are building, it is not necessary to construct a precise depth map, but to track image events in terms of qualitative transformations such as occlusion/disocclusion, detail emergence/disappearance. For this, approximate, object-based matching along the translational flowpaths suffices. Depending on sensitivity of this computation, exact stabilization may not be required, and inertial estimates can drift over time.

4. INFERENCE FOR NAVIGATION AND GUIDANCE

The first subsection defines different types of headings and the associated termination conditions. Section 4.2 uses the spatial representations of Section 2 and the heading structures to create algorithms that can guide a robot through the world based on visual information. The last subsection details rules that implement perceptually-based path planning and following, assuming there are no errors in re-acquiring landmarks, but not assuming that all landmarks are necessarily re-acquirable.

4.1 HEADING TYPES

A heading is defined to consist of:

- type
- sensor-centered coordinate system vector
- destination-goal
- direction-function
- termination-criteria

The type of a heading specifies the coordinate system that the direction conditions are computed in. A metric-heading-type corresponds to an absolute coordinate system. A metric heading can be induced by a correspondence of sensor position to a priori map or grid data, from an inertial sensor, geo-satellite location, dead-reckoning from a known initial position, etc. A viewframe-heading-type refers to headings computed between viewframes that share common visible landmarks. This corresponds to reasoning within a local landmark coordinate system. An orientation-heading-type is a coordinate-free heading based on observed relationships with LPB's. Note that crossing LPB's corresponds to entering new orientation regions.

Destination-goals are descriptions of places that the heading is intended to point the robot or vision system platform toward. A destination-goal may be specified as any combination of:

- a set of absolute (e.g., UTM) world coordinates
- a viewframe-localization
- an orientation-region
- a set of (simultaneously visible) landmarks

The types of destination-goals are listed above in order of increasingly topological, i.e., more qualitative, representation. Figure 2-6, shows the relationship between these levels of representation. A sensor-centered direction vector is a choice of immediate heading for robot locomotion. Range can be supplied if available.

Direction-functions accept runtime data at multiple levels of representation (i.e., orientation, view-based, and or metric) and return true if the heading is being maintained and false otherwise. Direction-functions are essentially predicates, except that they may have side-effects such as updating heading error parameters. Direction-functions for metric headings compare the desired heading vector against that returned from sensor readings. Direction for viewframe headings are given as relative angles between the heading vector in the sensor-based coordinate system and the observed landmarks recorded in viewframes. Finally, orientation directions are conjunctions of headings relative to LPB's. They specify a set of simultaneously true conditions indicating passage to the right, left, or between landmark pairs.

Termination criteria are runtime computable conditions that indicate that if the heading continues to be maintained, its direction-function can no longer return true. This can occur because the heading has been fulfilled, meaning that we have reached the destination goal implicit in the direction-function. For example, this occurs if we are at the desired absolute world coordinate location specified in a metric heading, if we recognize the set of landmarks and relative angles corresponding to a viewframe heading destination, or if we cross the LPB's given in the direction-function of an orientation heading. A heading will terminate with failure if we accumulate too much error in an absolute coordinate system tracking scheme or if we lose sight of the set of landmarks required to maintain a view-based or orientation-region heading. Termination criteria also include feedback conditions from modules outside the vision system, such as a path-planning module that reasons about obstacles, traversability of the ground surface, strategic concealment, etc.

Heading types, destination-goals, direction-functions and termination criteria are summarized in Table 4-1. The following explains how to compute direction-functions and termination criteria for basic inference at the viewframe and orientation region levels of spatial description.

Viewframe headings can be computed between two viewframes that share at least two landmarks (for 2D reasoning, three landmarks for 3D reasoning) in common. If the set of landmarks included in a viewframe destination are visible and we are very close to our original point of observation, then we can necessarily (up to traversability of the intervening terrain) perform a hill-climbing algorithm to bring the sensor to the point of observation where the viewframe was previously collected. This can be accomplished by dynamically computing a path based on control-feedback from the relative angles between the observed landmarks.

HEADING TYPE / FIELD	METRIC	VIEWFRAME	ORIENTATION
destination-goal	polygon expressed in absolute coordinate system	viewframe	orientation region
direction function	distance of current estimated point location to destination polygon	if sensor-centered vector-range set then, difference of current location to vector destination relative to point where heading set or maintain visibility of goal landmarks and hill-climb on relative angles	reduce LPB set to unique LPB and track until orientation reversal
termination criteria	estimated error in current location exceeds threshold or goal achieved	if range set then estimated error in relative location exceeds threshold or lose sight of landmarks in goal viewframe or hill climbing to relative angles fails or goal achieved	reduced LPB set is null or lose sight of goal LPB landmarks or must reverse a goal LPB to continue or goal achieved

Table 4-1: Heading Specifications

However, if we are outside the finest orientation region that contains our viewframe destination-goal, or if the goal is only to get to where a set of landmarks are all visible, then a hill climbing control-feedback approach will fail because we will have to cross LPB's to reach our destination-goal, and the relative angles between landmarks will vary non-monotonically.

An alternative approach is to formulate the viewframe localizations for each of the viewframes in the common local coordinate system defined by the two (or three) landmarks, and the sensor. Note that the the two landmarks must not be co-linear with the sensor (the three must not be co-planar with the sensor). We then have two regions expressed in the same coordinate system. Any affine linear transformation that maps one viewframe approximately onto the other may be taken as a heading. For example, we can translate the centroid of the first viewframe to the centroid of the second. This is the definition of viewframe heading transformation we use in this paper. It defines a heading as a vector pointing between the viewframes, and supplies the vision system with an intuitive notion of "head thataway". A viewframe heading is not generally the same as a metric heading, because the points in space that the sensor occupied when the viewframes were collected may not be mapped onto each other by the viewframe heading transformation.

Figure 4-1 illustrates the generic situation in which we can compute a viewframe heading. One viewframe contains the landmarks L_1 to L_6, and the other viewframe contains landmarks L_3 to L_9. Range estimates to L_3,L_4,L_5, and L_6 may be different in the two viewframes. Using the LPB connecting L_3 and L_4, we can assign a local orientation to the vector pointing from L_3 to L_4. This vector, with the implied orientation of the ground plane, defines a local 2D coordinate system.

Suppose that we have computed the viewframe localization for the first viewframe in terms of the coordinate frame defined by L_1 and L_2, and the second viewframe localization in terms of L_2 and L_3. Our problem is to determine the linear transformation that carries the L_1-L_2 coordinate system into the L_2-L_3 coordinate system. Because we have range intervals to the landmarks, this is an ambiguous computation. If the range from the sensor to the landmarks were fixed, the transformation for a point $[r_1,\phi]$ is given by:

$$[r_1,\phi] \longrightarrow [r_2,\pi+\theta_{12}+\phi-\alpha]$$

If we systematically vary the locations of L_1, L_2 and L_3 over their range estimate intervals, we obtain a continuous family of possible local coordinate transformations. However, because we intersect the transformed regions, the worst localization we can obtain is the best localization from any pair of landmarks.

Termination criteria for a viewframe-heading include that the destination goal is reached, that we have traveled the approximate distance to the goal predicted in the affine transformation, or that hill climbing fails.

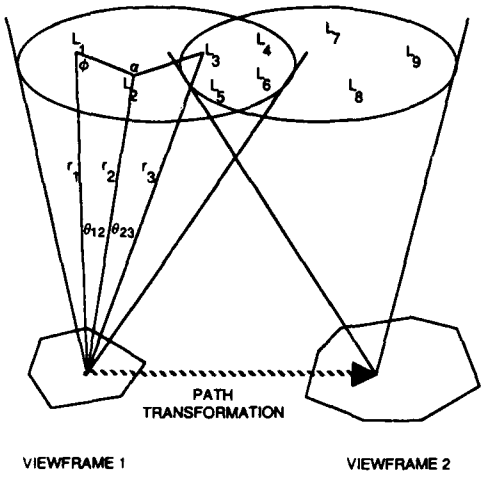


Figure 4-1: Viewframe Heading

Orientation-headings are conjunctions of specifiers for crossing LPB's. Recall that for the LPB formed from L_1 and L_2, i.e., LPB(L_1,L_2), the two sides are denoted $[L_1\ L_2]$ and $[L_2\ L_1]$ respectively. The landmarks L_1 and L_2 divide LPB(L_1,L_2) into three sections. If we cross LPB(L_1,L_2) from the $[L_1\ L_2]$ side, we can cross to the left of L_1, between the landmarks, or to the right of L_2. These crossings correspond to the visually observable events of L_1 occluding L_2 (or having identical azimuth angle in the sensor centered coordinate system), L_1 and L_2 being separated by 180 degrees, or L_2 occluding L_1. We denote these three possibilities by $l[L_1\ L_2]$, $b[L_1\ L_2]$ and $r[L_1\ L_2]$. Thus, for example, $l[L_1\ L_2]$ and $r[L_2\ L_1]$ are crossings of the same section of LPB(L_1,L_2), but in opposite directions. We also add the orientation-heading of moving toward a fixed landmark. We denote this heading with the symbol "a", and point out that we can place this heading in a uniform representation with the others by the artifice of using $a[L_1\ L_1]$ to mean "head toward landmark L_1". Note, of course, that L_1 cannot be crossed, but rather it is reached. We can view LPB attainment as reaching rather than crossing also, so this is not incompatible.

An orientation-destination-goal is a conjunction of LPB crossing specifiers, with no more than one crossing specifier per LPB in the conjunction. The basic termination condition corresponding to an orientation-heading is that all crossings have occurred. Termination can also occur if it is impossible to proceed without re-crossing an already crossed LPB, or if none of the LPB's are visible.

A top level orientation-heading consists of a goal to execute the crossings to get on the correct sides of the LPB's corresponding to visible landmarks that are also listed in the destination-goal. For example, suppose L_1, L_2, L_3 and L_4 are visible, and the destination-goal includes $[L_2\ L_1]$ AND $[L_4\ L_3]$. Then we must cross both LPB's. It is straightforward to select a single heading that will accomplish all crossings. For example, if the viewframe order is L_2, L_4, L_1, L_3, then a heading of $b[L_4\ L_1]$ will accomplish crossing both LPB(L_1,L_2) and LPB(L_3,L_4). A production system can be used to perform this reasoning about the order of landmarks in a viewframe to compute a heading-direction.

The usefulness of the production system is that it allows us to reduce a conjunction of LPB crossings to a single LPB crossing condition; if relevant, the production system deduces that the specified conjunction of headings is impossible.

The production system theory is based on a binary operation between pairs of orientation-headings. Because the operation is associative and commutative, the production system can be executed on a conjunction of orientation-headings, two at a time, in any order. Therefore, we can front the production system by a pre-processor that identifies the viewframe relative azimuth angle order of the set of landmarks involved in the orientation-heading conjunction. For a pair of orientation headings, there are six relative angle states that the four landmarks in the pair have. These are: that the landmarks of the two pairs are perceived in left to right order, the first two being the two from the first orientation-heading followed in azimuth angle by the two landmarks from the second orientation-heading; or that the landmark pairs "overlap", the first of the first pair being followed by the first of the second pair, then by the second of the first pair; and finally the second of the second pair; or, the landmarks of one pair could be nested between the landmarks of the other pair. These six possibilites, and the corresponding productions for combination of headings for two landmark pairs $[A\ B]$ and $[C\ D]$ are listed in Figure 4-2. Notice that many of the heading combinations are impossible.

		VIEWFRAME ORDER OF LANDMARKS					
[A,B] cross specifier	[C,D] cross specifier	ABCD	ACBD	ACDB	CABD	CDAB	CADB
r	r	r[C D]	r[B D]	r[D B]	r[B D]	r[A B]	r[D B]
r	l	b[B C]	--	--	--	--	--
r	b	b[C D]	b[B D]	--	b[B D]	--	--
r	a	a[C C]	--	--	--	--	--
l	r	--	--	--	--	b[D A]	--
l	l	l[A B]	l[A C]	l[A C]	l[C A]	l[C D]	l[C A]
l	b	--	--	--	b[C A]	b[C D]	b[C A]
l	a	--	--	--	--	a[D D]	--
b	r	--	--	b[D B]	--	b[A B]	b[D B]
b	l	b[A B]	b[A C]	b[A C]	--	--	--
b	b	--	b[C B]	b[C D]	b[A B]	--	b[A D]
b	a	--	--	a[C C]	--	--	--
a	r	--	--	--	--	a[A A]	--
a	l	a[A A]	--	--	--	--	--
a	b	--	--	--	a[A A]	--	--
a	a	--	--	--	--	--	--

Figure 4-2: Orientation Heading Binary Productions

The case when we are already on the goal side of an LPB requires more involved reasoning. We want to cross certain LPB's without crossing the ones we are already on the correct side of. Without (implicit or explicit) range information, we cannot tell which LPB we will cross first (on any heading); see Figure 2-5. One approach is to pick a heading corresponding to the conjunctive conditions in the orientation-destination-goal that we want to change, and dynamically estimate the rate of change of the relative angles between landmarks as the heading is maintained. A control-feedback approach can be used to estimate the order that LPB's will be crossed. Specifically, over a short period of time, typically less than a minute, we can project, based on the rate of change of the angle between the landmarks in an LPB, how long it will be before that angle reaches zero or 180 degrees, and adjust our heading accordingly.

4.2 LANDMARK-BASED ENVIRONMENTAL REPRESENTATION

A natural environmental representation based on viewframes recorded while following a path is given by two lists, one list of the ordered sequence of viewframes collected on the path, and another of the set of landmarks observed on the path. We call the viewframe list a viewpath. The landmark list acts as an index into the viewpath, each landmark pointing at the observations of itself in the viewframes. For efficiency, the landmark list can be formed as a database that can be accessed based on spatial and/or visual proximity. Visual proximity can be observed, or computed from an underlying elevation grid and a model of sensor and vision system resolution.

Recall that the primary component of an orientation-region-heading is a conjunction of LPB passage specifiers. Therefore we use an environmental representation for orientation-region reasoning that is a list of oriented LPB's encountered and crossed in the course of following a path. We call such a list an orientation-path. As with viewpaths, there is an associated landmark list that indexes into the orientation-path.

A dynamically acquirable environmental representation that merges the representations for viewpaths and orientation-paths consists of an ordered list interspersing viewframes, LPB crossings, and appearance and occlusion (or loss of resolution) of landmarks, as well as recording the headings taken in the course of following the path over which the environmental map is being built. Thus, we can integrate the representations required for viewframe and orientation region based reasoning with heading and landmark information to formulate an environmental representation that supports hybrid strategies for navigation and guidance. The representation is formed at runtime and consists of multiple interlocking lists of sequential, time-ordered, lists of visual events that include those necessary for the algorithms presented in Section 4.3.

A landmark list is separately maintained. It encodes proximity information between landmarks, and acts as an index into the dynamic environmental path representations. The data structure of the environmental representation is pictured in Figure 4-3.

The first occurrence of a landmark points at the instantiated schema in the vision system database that was used to gather evidence in the landmark recognition process. After that, all recognized re-occurrences of this landmark point back at this initial instance. The same is true for the first occurrences and successful re-recognition of LPB's and viewframes. This mechanism allows multiple visual path representations, built at different times, to be incrementally integrated together as they are acquired by using a common landmark indexing/pointer list.

```
;; Viewpaths and sighting events

(defstruct viewpath
        date-initiated
        time-initiated
        start-location
        initial-destination-goal
        event-sequence-vector)

(defstruct viewpaths-event-id
        viewpaths-occurs-on
        number-of-event-on-viewpath
        time-of-observation
        original-event-instance
        first-sighting-since-last-disappearance
        previous-occurrence
        next-occurrence)

(defstruct (viewframe
            (:include viewpaths-event-id))
        ordered-landmark-sightings
        range-intervals
        relative-solid-angles
        relative-angular-errors
        relative-azimuth-angles
        ordered-orientation-reversals)

(defstruct (original-landmark-instance-event
            (:include viewpaths-event-id))
        landmark-name
        schema-instantiation)

(defstruct (landmark-sighting-event
            (:include viewpaths-event-id))
        landmark-name)

(defstruct (landmark-disappearance-event
            (:include viewpaths-event-id)))
        landmark-name)

(defstruct (lpb-sighting-event
            (:include viewpaths-event-id))
        lpb-name)

(defstruct (lpb-disappearance-event
            (:include viewpaths-event-id))
        lpb-name)

(defstruct (lpb-crossing-event
            (:include viewpaths-event-id))
        lpb
        crossing-specifier)
```

Figure 4-3: Environmental Representation Data Structure

4.3 VISUAL PATH PLANNING AND FOLLOWING

The top level loop for landmark-based path planning and following is to:

1) determine a destination-goal

2) compute and select a current heading

3) execute the heading while building up an environmental representation

The destination-goals are typically determined recursively, implementing a recursive goal-decomposition approach to perceptual path planning.

In this section we give two algorithms, presented as sets of rules, for performing this top level loop for the cases of viewframe-based path following and for path constraints through orientation nets. Note that algorithms for path planning and following over metric representations already exist; see, for example, [Linden et.al. - 86]. A rule-based implementation of these algorithms permits hybrid perceptual path planning and following strategies, that can respond opportunistically to the available data and the certainty of the vision system output.

Inference over viewframes performs path planning and following over a visual memory, and therefore assumes that viewpaths, i.e., the visual memory, have already been collected. If they have not, paths must be planned and followed based on metric data and viewpaths collected in the course of following those paths.

The concept underlying the path planning following strategies encoded in these rules is to mix the following approaches as knowledge is available or can be inferred:

- find landmarks in common between viewframes between point of origin and viewframe-destination and compute vector (i.e., direction and approximate range) headings between viewframes
- locate and get on the correct side of LPB's specified in an orientation-destination, or
- associate visible and goal landmarks with map data and compute a metric heading between current location and goal

In the following we present several rules that effect these strategies and then develop some rules that mix the approaches in attempting to reason opportunistically. Notice that each of these strategies provably reaches its goal, up to the perceptual re-acquisition of landmarks and the traversability of intervening terrain.

```
if viewframe goal landmarks visible
  --> compute viewframe-heading
      if at least one LPB has an incorrect orientation
          relative to our viewframe-destination-goal
      then follow heading for approximate distance
          estimated by the viewframe-heading
      else maintain heading by control-feedback
          path following on relative angles between
          landmarks
      build a new viewpath to destination goal, using
          the existing landmark list where possible

if viewframe goal landmarks not visible and viewpaths
exists
  --> make a viewframe of the currently visible region
      chain back through viewpaths until common
          landmarks are located
      chain forward through viewframes setting up
          intermediate destination-goals
      recursively execute viewframe headings to reach
          the destination goals corresponding to
          visible landmarks

if viewframe goal landmarks not visible and no
viewpath exists
  --> set goal to find a metric heading
```

The advantages of orientation region based reasoning over viewframe reasoning are that orientation-headings require less computation, and LPB's can be accurately acquired while moving at high speed relative to the processing power of the vision system. The latter is because we do not have to include the relative angles between landmarks in LPB's; so if we extract a viewframe from buffered images, this point does not apply.

LPB's are visually observable one-dimensional subspaces of the ground surface. Therefore, if we record a viewframe each time we cross an LPB (that we are tracking), we should be able to re-acquire that viewframe by searching along the LPB. We can use control-feedback on the relative angles between landmarks to locate the viewframe again. Being one-dimensional, this search is very efficient.

Inference over orientation-paths, analagous with viewpaths, can be performed in a visual-memory of LPB's and landmarks. However, orientation-paths can also be approximately inferred from viewpaths. If two viewpaths were sequentially acquired, then all boundary orientation reversals between the two viewframes must have occurred on the path between the viewframes. Although we cannot order them without metric data, we can still use this partially ordered sequence of LPB crossings to reason over; in particular we can use the production system for reduction of conjunctions of headings to compute an orientation-heading between viewframes that we have already collected.

if orientation region goal landmarks visible
--> compute orientation-heading
control feedback on LPB, using match to viewframe along boundaries to locate missing, but predicted landmarks
build a new orientation-path to goal, using existing landmark list

if orientation region goal landmarks not visible and orientation-path exists
--> chain back through orientation-paths until landmarks common with current viewframe are located
chain forward through boundary crossings setting up orientation-destination-subgoals recursively
execute orientation-headings to reach the destination goals corresponding to visible landmarks

if orientation region goal landmarks not visible and no unexplored orientation-path exists
--> set goal to find metric heading

In the following we present detail on methods for tracking back through the environmental representation to find landmarks that either are common with both the current viewframe and the destination-goal landmark set, and for computing a plan as an ordered list of of recursively executable sub-goals.

The key to the search is to use the landmark database as a random access into the criss-crossing one-dimensional path structures gathered on previous system runs. If goals are given for regions in which there are no environmental maps collected, goals are set up to develop headings from inference in metric data; however, the metric (e.g., map) data is not entered into the environmental representations until it is actually perceived. We present the search methods as a set of rules that operate over the environmental representation and set up goal structures and pointers into it.

if goal to chain current viewframe to viewframe-destination-goal
--> if can metrically localize destination-goal viewframe
--> match viewframe against landmark-database access for that metric localization
for each landmark in current-viewframe intersect pointers to LPB's and viewframes with set of LPB and viewframe pointers for all landmarks in destination set
if any intersections are non-empty
--> set destination sub-goal to reach the LPB or viewframe in the intersection
else chain from destination through environmental paths until paths are exhausted or a landmark, LPB or viewframe with current-viewframe landmarks in it is found and set up a goal to follow the path
else match viewframe against entire landmark-database and repeat above with best viewframe matches

if goal to metrically localize a viewframe
--> if viewframe has grid attachment
--> use it
else check if any landmarks in viewframe have grid attachments and average them
else attempt to make grid attachment for viewframe by matching landmarks in map data with line-of-sight reasoning for elevation
else chain back along paths from viewframe to last headings before viewframe was collected and see if they had metric headings and set sub-goals to reach them
else return failure to metrically localize

if goal to chain back from metric-destination and no viewpath or orientation-path chain can be found
--> set metric-heading until range reached or obstacle encountered

if metric-heading fails due to obstacle and viewpaths or orientation paths cannot be found from the failure point
--> backtrack to skirt obstacle in direction of heading

if goal to skirt obstacle with metric heading failure
--> if can match obstacle in map
--> plan passage on map and set up sequence of metric headings with visual and range termination conditions
else set skirt-heading and set daemons to find passage beyond obstacle

5. RESULTS

Several simulations of landmark acquisition scenarios and extracted viewframes, orientation regions, viewframe localizations and viewpaths from them have been implemented. Figure 5-1 shows the simulated landmark data over which viewframe localizations have been extracted. Figure 5-1(a) shows an image take at the Martin Marietta Autonomous Land Vehicle development site. Figures 5-1(b)-(e) show various displays of the 30 meter U.S. Army Engineer Topographic Laboratories terrain grid data gathered over the area in the image. Figures 5-1(b) and (e) show two different perspectives on a wire-frame view of the elevation data. The displays include paint for terrain type overlay, and a building present on the site. Figure 5-1(c) is an orthographic representation of the same grid data, while Figure 5-1(e) is a painted perspective display. Here the coarse quantization of the road area in the foreground is evident. The circles on Figure 5-1(c) indicate three manually selected landmark points representing two peaks and the building. The x-ed circle on the right is the location of the simulated sensor.

Figures 5-2(a)-(f) show perspective and orthographic views of the viewframe localizations obtained relative to pairs of landmarks. Range intervals of 50% the true range were used. Because the landmarks are approximately 50 pixels from the sensor, this corresponds to range intervals 800 meters long for landmarks 1600 meters away. Angular errors of .1 radian were used. In a 45 degree field of view for an image 512 pixels wide, this is approximately a 65 pixel error.

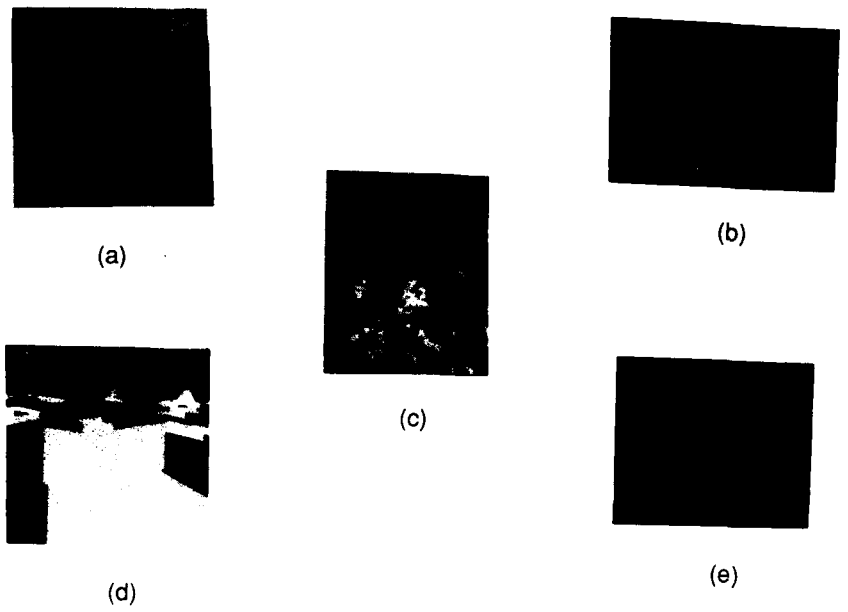


Figure 5-1: Landmark Data

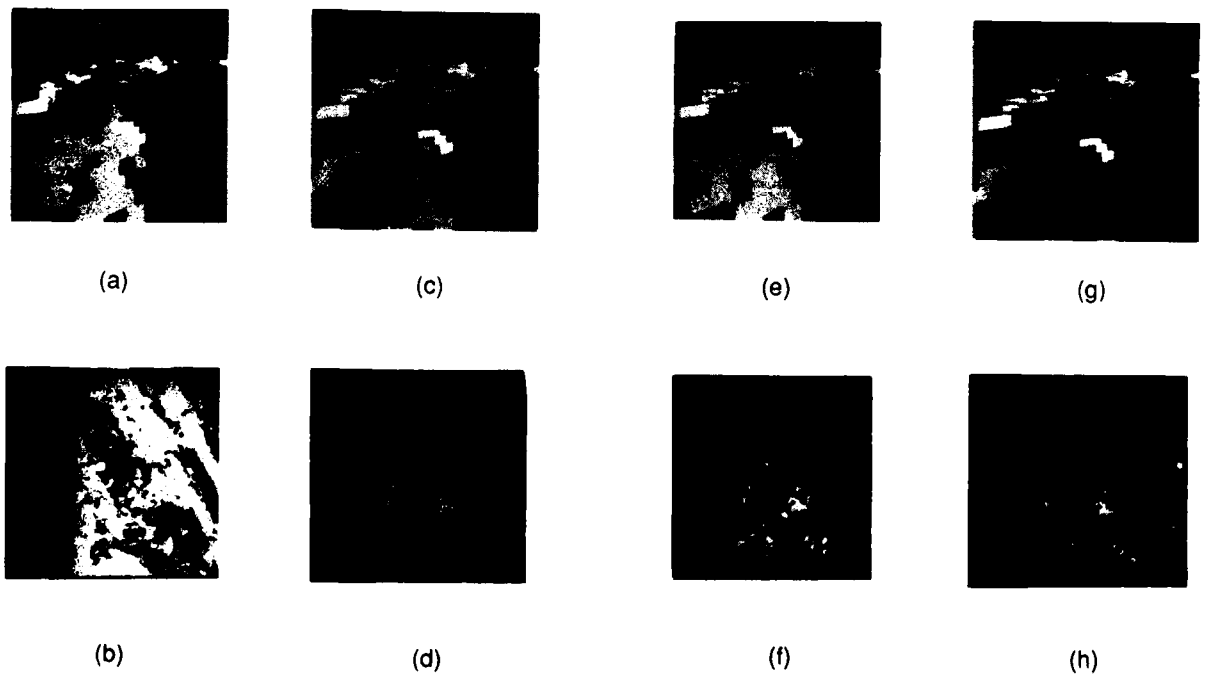
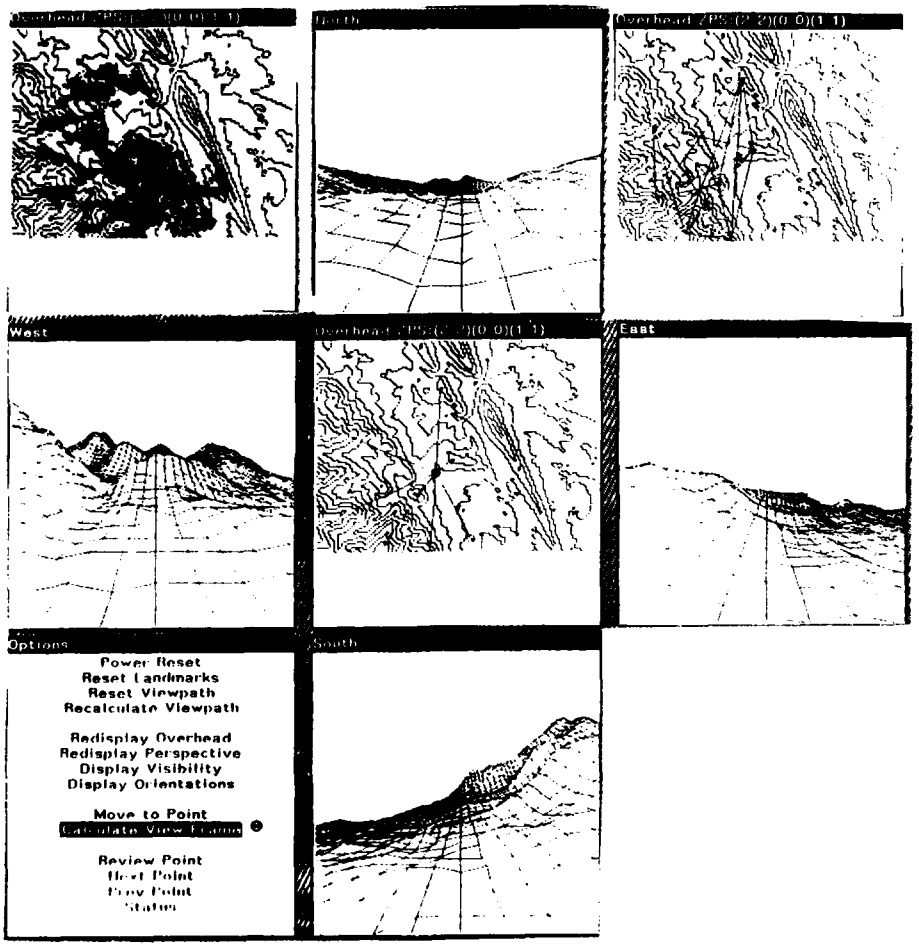


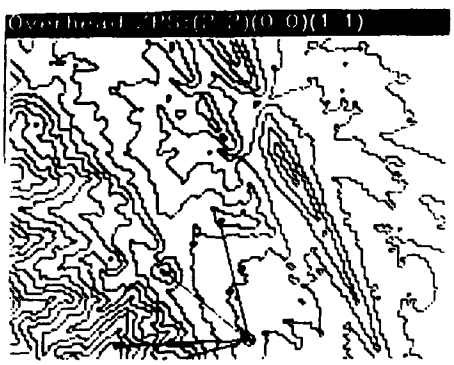
Figure 5-2: Viewframe Localization



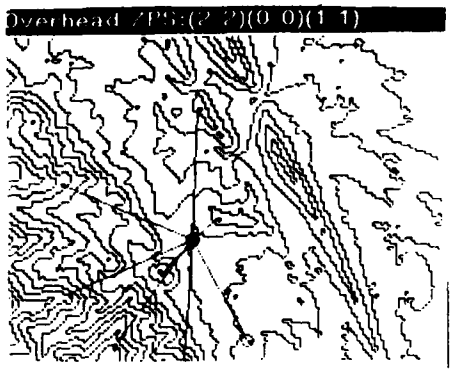
LISP Listener

```
Number Landmarks: 11
Number Viewframes: 4
Current Position (31,57,110)
Landmarks: (<LM: NIL (57,68,131)> <LM: NIL (62,71,138)>
<LM: NIL (39,90,136)> <LM: NIL (5,55,131)> <LM: NIL (29,
10,190)> <LM: NIL (50,19,195)> <LM: NIL (80,53,150)>)
Azimuths: (0.023953736 0.9087661 1.8854042 1.4514971 0.5
0617504 1.0256968 -5.8014927)
Elevations: (0.87625897 -0.06693965593446904d0 -0.102112
11839784795d0 0.85799766 0.21544746 -0.7815998236285608d
0 -0.6052954832660l2d0)
Ranges: (35.185223 44.05678 42.76681 33.48134 92.80625 9
5.02631 83.379807)
```

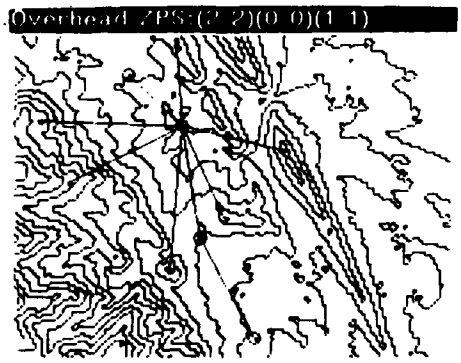
Figure 5-3: Viewpath Simulator



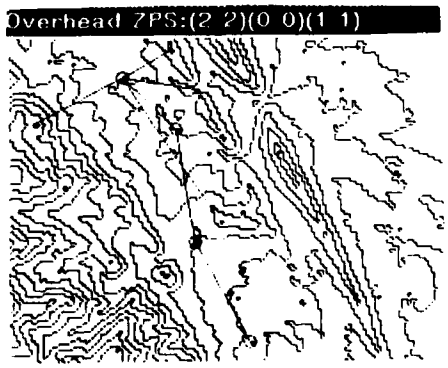
(a)



(b)



(c)



(d)

Figure 5-4: Viewpath Execution

Both errors are far greater than we expect in practice. The intersection of the landmark pair localizations, resulting in the viewframe localization, is shown in Figures 5-2(g) and (h).

Figure 5-3 shows an example from the viewpath simulator. A 360 degree sensor is assumed. Perspective windows are used to manually select landmarks. The overhead windows are used to pick a path based on output from the navigation and guidance inference rules. Visibility (upper left), viewframes, (center), and orientation regions (upper right), are calculated as the path is executed. The LISP listener shows the output from the simulator for the viewframe pictured above it. Although the simulator outputs absolute ranges to landmarks, they are altered to coarse intervals or eliminated before being passed to the inference process.

Figure 5-4 shows the results of executing a series of orientation headings to create a viewpath through the elevation data. At each point it is assumed that all visible landmarks are re-acquired and correctly associated to prior occurrences.

6. SUMMARY AND FUTURE WORK

A rigorous theory of qualitative, landmark-based navigation and guidance for a mobile robot has been developed. It is based upon a theory of representation of spatial relationships between visual events that smoothly integrates topological, interval-based, and metric information. The rule-based inference processes opportunistically plan and execute routes using visual memory and whatever data is currently available from visual recognition, range estimates and a priori map or other metric data.

Key on-going development tasks include:

- integration of the navigation and guidance capability with the vision system, and
- addition of a non-monotonic reasoning system that accounts for imperfect re-acquisition of landmarks.

Other tasks include reasoning with extended landmarks, such as road, rivers and ridges, and inference of shape and additional topological relationships such as region containment.

ACKNOWLEDGEMENTS

This document was prepared by Advanced Decision Systems (ADS) of Mountain View, California, under U.S. Government contract number DACA76-85-C-0005 for the U.S. Army Engineer Topographic Laboratories (ETL), Fort Belvoir, Virginia, and the Defense Advanced Research Projects Agency (DARPA), Arlington, Virginia.

The authors wish to thank Angela Erickson for providing administration, coordination, and document preparation support.

REFERENCES

[Bajcsy et.al. - 86] - R. Bajcsy, E. Krotkov, and M. Mintz, "Models of Errors and Mistakes in Machine Perception", University of Pennsylvania, Computer and Info. Science Technical Report, MS-CIS-86-26, GRASP LAB 64, 1986.

[Bolles and Baker - 85] - R. Bolles and H. Baker, "Epipolar-plane Image Analysis: A Technique for Analyzing Motion Sequences", Proceedings for the Third Workshop on Computer Vision: Representation and Control, October 13-16, 1985, pp. 168-178.

[Cao et.al. - 86] - Z. Cao, S. Oh, and E. Hall, "Dynamic Omnidirectional Vision for Mobile Robots", Journal of Robotic Systems, Vol. 3, No. 1, 1986, pp. 5-17.

[Davis - 86] - E. Davis, "Representing and Acquiring Geographic Knowledge", Courant Institute of Mathematical Sciences, New York University, Morgan Kaufmann Publishers, Inc., 1986.

[Eastman and Waxman - 85] - R. Eastman and A. Waxman, "Disparity Functionals and Stereo Vision", Proceedings Image Understanding Workshop, Miami Beach, Florida, December 9-10, 1985, pp. 245-254.

[Hebert and Kanade - 85] - M. Hebert and T. Kanade, "First Results on Outdoor Scene Analysis Using Range Data", Proceedings Image Understanding Workshop, Miami Beach, Florida, December 9-10, 1985, pp. 224-231.

[Kuipers - 77] - B. Kuipers, "Representing Knowledge of Large-Scale Space", Massachusetts Institute of Technology, Artificial Intelligence Laboratory Technical Report, AI-TR-418, July 1977.

[Lawton - 83] - D. Lawton, "Processing Translational Motion Sequences", Computer Vision, Graphics, and Image Processing, Vol. 22, 1983, pp. 116-144.

[Lawton et.al. - 86] - D. Lawton, T. Levitt, C. McConnell, and J. Glicksman, "Terrain Models for an Autonomous Land Vehicle", IEEE International Conference on Robotics and Automation, San Francisco, California, April 1986.

[Linden et.al., - 86] - T. Linden, J. Glicksman, K. Dove, and Y. Kanayama, "DARPA Autonomous Land Vehicle Project Semi-Annual Report to Martin Marietta Corporation", TR 1085-14, Advanced Decision Systems, Mountain View, California, October 1986.

[Lucas and Kanade - 84] - B. Lucas and T. Kanade, "Optical Navigation by the Method of Differences", Proceedings Image Understanding Workshop, New Orleans, Louisiana, October 3-4, 1984, pp. 272-281.

[Matthies and Shafer - 86] - L. Matthies and S. Shafer, "Error Modelling in Stereo Navigation", Carnegie-Mellon University, Computer Science Department, Technical Report, CMU-CS-86-140, 1986.

[McDermott and Davis - 84] - D. McDermott and E. Davis, "Planning Routes through Uncertain Territory", Artificial Intelligence - An International Journal, Vol. 22, No. 2, March 1984, pp. 107-156

[Massey - 67] - R. Massey, "Introduction to Algebraic Topology", Addison-Wesley, 1967.

[Moravec - 80] - H. Moravec, "Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover", Ph.D. Thesis, Stanford University, September 1980.

[Pentland - 85] - A. Pentland, "A New Sense for Depth of Field", Proceedings of the Ninth International Joint Conference on Artificial Intelligence, IJCAI-85, Los Angeles, California, August 18-23, 1985, pp. 988-994.

[Schone - 84] - H. Schone, "Spatial Orientation - The Spatial Control of Behavior in Animals and Man", Princeton Series in Neurobiology and Behavior, R. Capranica, P. Marler, and N. Adler (Eds.), 1984.

[Smith and Cheeseman - 85] - R. Smith and P. Cheeseman, "On the Representation and Estimation of Spatial Uncertainty", SRI International Robotics Laboratory Technical Paper, Grant ECS-8200615, September 1985.

Interpretation of Terrain Using Hierarchical Symbolic Grouping for Multi-Spectral Images

Bir Bhanu and Peter Symosek

Honeywell Systems & Research Center
3660 Technology Drive, Minneapolis, MN 55418

ABSTRACT

An Autonomous Land Vehicle (ALV) must be able to navigate through terrain in order to accomplish its mission of surveillance, search and rescue and munitions deployment. To reliably label each region seen in an image, the vision algorithm must make use of as much a priori information as possible because of the immense variability of outdoor scenes. The a priori information may be of many types, for instance, the nominal location of each terrain class in the image as a function of the imaging system's orientation. A knowledge-based algorithm for terrain interpretation for multi-spectral imagery is described. This is a significantly new approach to this task because, unlike the classical tree classifier approaches used in most of the remote sensing literature, only knowledge-based techniques are used to classify image regions. This permits accurate classification of the regions, despite imprecise a priori information for the feature values. The algorithm's design is described along with a discussion of the new research issues which have been identified from analysis of the algorithm's performance for real multi-spectral images.

1. INTRODUCTION

The interpretation of terrain is of critical importance to the Autonomous Land Vehicle (ALV), if it is to survive and carry out its mission in a totally unstructured natural environment. In order to perform the functions of obstacle avoidance, surveillance and path planning, the ALV must first interpret the imagery obtained from its sensors into various regions in terms of their land cover, trafficability and slope. The challenges presented by this task are significantly greater than those encountered for navigation on paved highways because of the great variability of the terrain. Therefore, statistical approaches to the image interpretation problem for this case are not sufficiently robust without adaptive feature detection and world knowledge.

Knowledge-based techniques for region labeling can tolerate large errors in feature data and still produce meaningful results. They can be designed in stages because of their modular rule database: as new contexts are discovered for classification of features, the system is reconfigured by the definition or modification of a few rules. Knowledge-based systems are very efficient for the task of performing retrieval operations on large symbolic databases on the basis of relational and contextual constraints on the data. It is this critical capability that makes knowledge-based techniques so useful for terrain interpretation: The contextual information that is applicable to all natural environments from a general world model can be used to improve the algorithm's performance and reliability.

A substantial amount of work has been done on the problem of labeling regions in segmented images with knowledge-based techniques. These systems have been used for photointerpretation applications,[1-3] autonomous weapon delivery systems,[4] and the labeling of features in arbitrary urban scenes.[5-7] These expert systems have several features in common: A database of calculated image features is matched with predicates of production rules, which are represented as logical statements of the form "If ...., then ...." and a control system that supervises rule activation.

The system developed by Nagao and Matsuyama[2] uses a knowledge base representing relational, contextual and geometric constraints for the task of region labeling for multi-spectral imagery obtained from low-flying aircraft. The region boundaries are detected by a variety of low-level image segmentation algorithms and the resultant information is archived on a blackboard shared by each of the experts of the system. Each expert is optimized for locating a specific kind of object or region. They devised an approach for the reliable classification of vegetational regions that is independent of the time of year, using the ratio of two distinct spectral bands to discriminate the vegetation regions from the non-vegetation regions. They demonstrate that knowledge-based techniques permit the reliable identification of houses and roads in congested urban scenes where other classical approaches normally fail. The approach that they use is essentially hierarchical because the segmentation and classification of micro-level regions of the image, such as houses or cars, is dependent on the preliminary segmentation of the region that contains them. A region growing approach is used to identify large regions of the image with global region properties such as homogeneity of multi-spectral intensity levels or elementary texture measures, but no texture information is used at this stage, which is critical for the detection of regions in arbitrary outdoor scenes. Also, they do not use any map information, which is very useful for the segmentation of natural scenes.

Ohta[6] developed a hierarchical region labeling

scheme for color images of urban scenes. The approach is hierarchical because an initial plan image is derived and labeled before a more detailed, data-directed segmentation is carried out. The plan image is defined by a region-based color image segmentation algorithm. The macro-level regions of the plan image are: sky, tree, building and road. These region categories are detected using top-down contextual and spectral constraints. The algorithm is very reliable and can correctly label regions in urban outdoor scenes using only 57 rules. His algorithm employs a texture measure for region classification, but the measure is expressed as a three valued logical variable: a region is either not-textured, textured or heavily-textured. This approach is useful for the discrimination of tree regions from sky, buildings and roads, which are essentially untextured, but the approach is not sufficiently robust for the discrimination of the textures found in terrain images.

Previous work on the problem of scene labeling for remote sensing applications has predominantly been restricted to classical techniques. Landgrebe[8] and Swain[9] surveyed the state-of-the-art for this application and identified four generic approaches to this problem: 1) Spectral Methods, 2) Spectral/Temporal Methods, 3) Spectral/Spatial Methods and 4) Spectral/General Scene Context Methods. Algorithms from category 4) do employ auxiliary information, such as digital map data, water table depth maps and ownership boundaries, but no work has been described in the literature on the value of knowledge-based techniques for the classification of regions in remotely-sensed imagery. Region labeling with a priori spatial constraints is accomplished with relaxation labeling for consistency,[10] and measures of image texture. Image texture may be measured locally with techniques such as the co-occurrence matrix[11, 12] or globally, with techniques such as production system grammers for image structure.[13] The major obstacle to defining a knowledge base for the interpretation of remotely sensed imagery is that it is exceedingly difficult to identify structural rules, for this kind of imagery, that are universally valid.[9]

Goldberg, et al[14] designed an expert system for the detection of changes in the forests of Newfoundland from LANDSAT imagery. The system is hierarchical, with multiple tiers of experts applying domain knowledge, in a top-down fashion, to the problems of detecting foresting, forest fire damage and disease infestation. The experts exchange information for control and hypothesis generation on a blackboard message space. Knowledge-based techniques are used at the intermediate level of the hierarchy for the tasks of change detection and the interpretation of changes in forestry terms, but no knowledge-based systems are used for region classification. The results produced by these knowledge sources are extremely useful.

Other researchers have evaluated Markov Random Field (MRF) representations for image texture for the task of multi-spectral image segmentation.[15] A Markov Random Field is a compact representation of the inter-dependencies of the multi-spectral intensities of pixels of a neighborhood region. These models are very effective at distinguishing between regions whose means are the same, but whose textures are different.

Our approach is modeled after the urban scene interpretation algorithms referred to previously.[5-7] What distinguishes our work from this previous work is the use of context-dependent constraints in the knowledge base, such as map data or features cues from a Landmark Recognition/Prediction algorithm.[16] Constraints imposed on the region labeling process with a priori information for the geographic region where the vehicle is traveling improve the reliability of the knowledge-based system significantly. This approach will play a vital role in the development of a totally autonomous system, where the vehicle will be able to navigate without external guidance through an unstructured natural environment.

Interpretation algorithms for remote sensing applications often encounter problems caused by insufficient spatial resolution of the sensor. The effect of insufficient spatial resolution is that the spectral response observed at one location (generally a pixel) is a mixture of the spectral responses of several individual land categories, because the geographical area subtended by a pixel contains a mixture of terrain.[8] This problem is called the mixture-pixel problem. For the ground-to-ground ALV scenario, a similar problem exists. For this scenario, the resolution is, in fact, too high for pixel-based labeling. In general, the spectral response of an arbitrary region of a multi-spectral image will not be equivalent to the spectral signature of a single category of terrain, because observed regions will be composed of multiple vegetation and land cover categories. Therefore, the signature of regions seen in multi-spectral images will represent the accumulated responses of several vegetational and geological features. Because of the high resolution of the images, region-based calculations are required to defeat the effects of sensor noise. However, region-based measurements will be distorted by the inhomogeneity of the data. Constraints derived from context-dependent knowledge can resolve this dilemma. Expert systems are extremely efficient at defining the target set of several symbolic constraints applied to a relational database. Thus, knowledge-based systems are the most appropriate algorithm for terrain interpretation in terms of efficiency and generality.

The knowledge-based methodology, that we have labeled Hierarchical Symbolic Grouping for Multi-spectral data (HSGM), is an innovation in this field for several reasons. The approach is hierarchical and, therefore, is robust despite significant discrepancies between prior information in the knowledge base and the actual image. The first stage of the algorithm is the segmentation of the image into macro-level regions with a gradient-based technique that is optimal for the spectral characteristics of the terrain categories that the algorithm is designed to detect. The macro-level regions are: sky, forest, field, and road. The algorithm defines only closed region boundaries, and these closed regions are classified to one of the four categories with a knowledge-based approach which uses relational, locational and spectral constraints. The macro-level regions are further segmented with the gradient-based approach. For this stage of the hierarchy, the gradient-based algorithm's parameters are the

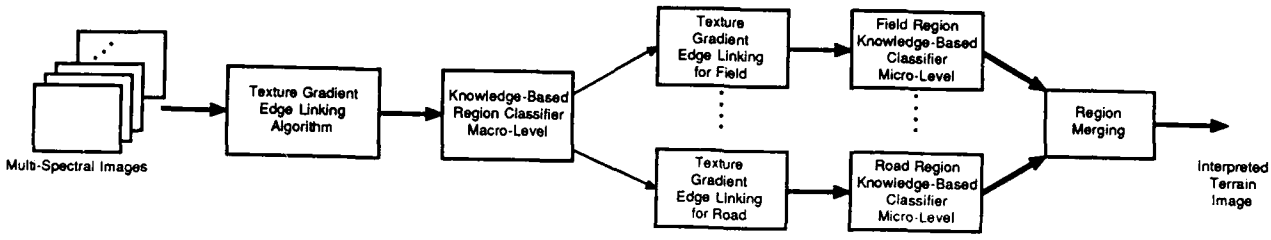


Figure 1. Hierarchical Symbolic Grouping for Multi-Spectral images (HSGM) algorithm block diagram.

optimal parameters for the individual macro-level region. Then, in the next stage, the resulting subregions are classified with a knowledge source that can use contextual constraints for the scene's structure. Because the knowledge base for each macro-level region contains information only for the terrain category it classifies, the system's complexity is greatly reduced.

The algorithm's design will be described in the next section. The principles of the design of the gradient-based region segmentation algorithm and the approach to knowledge-based classification of regions are discussed. Experimental results obtained for real ALV Multi-Spectral Scanner imagery are discussed in Section III. The terrain interpretation algorithm is still undergoing revisions as new procedures for fine-tuning its performance become apparent. The current status of the algorithm will be explained. Section IV itemizes the important features of this algorithm and the contributions made to the state-of-the-art of natural scene interpretation. The potential for further development of this algorithm for detailed terrain analysis for all categories of natural terrain is also discussed.

2. CONCEPTUAL APPROACH

2.1 Problem Statement

The terrain interpretation algorithm is designed to label regions seen in a single frame of multi-spectral imagery to one of several land cover categories, possibly including information regarding the slope, trafficability and geological characteristics of the region. Examples of labels which may be assigned to regions are field with grass and chokeberry vegetation or forest with hemlock vegetation located on class C terrain. It may not be possible to label a region to a single category, but the candidate set for the land cover should be specific enough for a navigation system to either match the region to a digital map database or to plan a course across the terrain without hitting any obstacles.

2.2 HSGM System Overview

The terrain interpretation algorithm is subdivided into five stages which are:

1. *Generation of "Plan" Image* - The segmentation of the multi-spectral image into a "plan" image.
2. *Knowledge-Based Classification, Macro-Level* - The classification of the regions of the plan image with a knowledge-based scheme.
3. *Macro-Level Region Segmentation* - The segmentation of the labeled macro-level regions with gradient-based techniques which are specifically designed for those regions.
4. *Knowledge-Based Classification, Micro-Level* - The labeling of subregions with contextual and spectral knowledge.
5. *Conflict Resolution* - The revision of "plan" image region boundaries if subregions are found at their borders that do not belong in that macro-level region.

The block diagram of the HSGM algorithm is shown in Figure 1.

2.3 Algorithm Details

In the following paragraphs, the algorithms that form each of the five stages will be described in detail. For each of the algorithms, the features that distinguish our approach from classical techniques will be pointed out.

2.3.1 Stage 1. Generation of "Plan" Image - As stated previously, the algorithms of this stage define a "plan" image which is the basis of all succeeding region classifications. The "plan" image is a coarse representation of the image because the algorithms' parameters are tuned to detect only larger features. Since the area of these regions is large, they will suffer very little degradation from sensor noise or from invalid algorithm parameters and can be extracted reliably from the image.[6] The calculation of the "plan" image is also a very important data formatting step, transforming the raw image data into a symbolic form which is more easily used by a high-level, knowledge-based labeling algorithm. We have named this innovative new approach *Texture-Gradient Edge Linking/Relaxation*. A block diagram of the approach is shown in Figure 2. Each of its constituent algorithms will now be explained.

The Texture-Gradient Edge Linking/Relaxation approach is more robust than traditional gradient-based techniques for the following reasons: For the traditional gradient-based approach, a gradient image is calculated with a differential window operator. The resultant gradient image is transformed into a binary image with an appropriately chosen threshold. If the value chosen for the threshold is too low, the resultant region boundary image will be degraded by noise boundaries, output along with the true

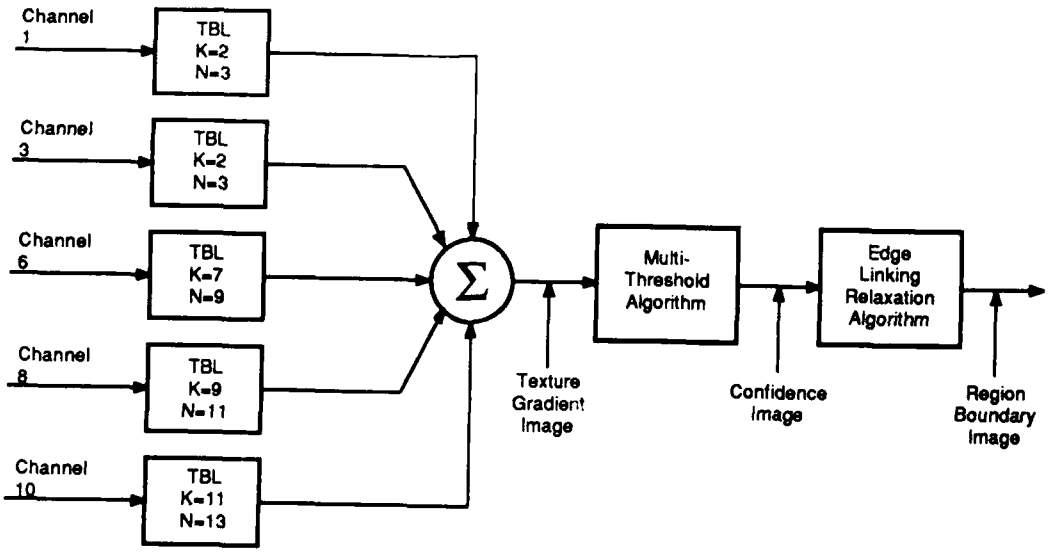


Figure 2. Texture-Gradient Edge Linking Relaxation algorithm. TBL is the Texture Boundary Locator, K and N are parameters used in the Texture Boundary Locator algorithm.

region boundaries. If the value chosen is too high, the region boundaries will be degraded by many gaps. One of the algorithms of stage 1, the Multi-Spectral Edge Linking Relaxation algorithm, uses multiples thresholds for the estimation of the true region boundaries. It endeavors to fill in the gaps of the highest threshold binary image by using the evidence from the edge points that are output for each of the remaining thresholds in an optimal fashion. In this way, the confidence that the edges of the output binary image are true region boundaries is high.

The gradient operators employed by the algorithm do not have to be intensity gradient operators: The only requirement made on the gradient image is that it should have a large magnitude where major discontinuities occur between regions of interest in the image. Because the edges between most regions in natural scenes are gradual transitions from one region's mean intensity to the neighboring region's mean intensity, we chose to use the Texture Boundary Locator algorithm to define the gradient images.

This algorithm calculates the texture gradient of an image, which is the local rate of change of a textural attribute of the image. The textural attribute is derived from the mean μ and standard deviation σ of each N x N window of the image, where N is obtained as a function of the image features to be detected.

The texture gradient is calculated as follows: A 2K+1 by 2K+1 window is centered at each pixel, where K is a function of the size of the region of interest and/or range information. The window geometry for an arbitrary image plane location P is shown in Figure 3. The pixels at the centers of the four sides and the corners of the 2K+1 by 2K+1 window are labeled sequentially beginning at the top left corner as shown in the figure. The texture gradient is obtained as:

$$\max_{0 \le i \le 3} \left\{ (\mu_i - \mu_{i+3})^2 + (\sigma_i - \sigma_{i+3})^2 \right\}^{\frac{1}{2}} \qquad (1)$$

The texture gradient is calculated only for those multispectral images which display sharp differences between the means of regions representing the four macro-level classes: sky, forest, field, and road. As shown in figure 2, multispectral channels 1, 3, 6, 8, and 10 are the best images of the 12 for this purpose.

The criteria employed to select these channels will now be explained. Multi-Spectral Scanner data is collected in 12 discrete bands, which are listed in Table 1.[17] The spectral characteristics of each of the regions of interest can be verified visually from the images themselves or from spectral reflectance measurements, such as those distributed by the U.S. Army Engineering Topographic Laboratories

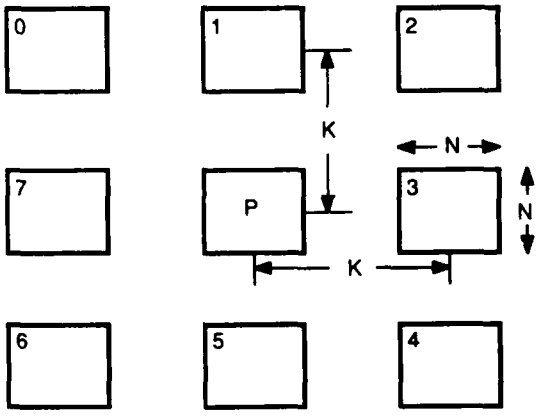


Figure 3. Texture gradient window geometry for Texture Boundary Locator algorithm.

Channel	Spectral Band (microns)
1	.44 - .49
2	.49 - .54
3	.54 - .58
4	.58 - .62
5	.62 - .66
6	.66 - .70
7	.70 - .74
8	.77 - .86
9	.97 - 1.06
10	1.0 - 1.4
11	1.5 - 1.8
12	2.0 - 2.6

Table 1. Multi-Spectral Scanner Channels

-	Sky	Forest	Field	Road
Sky	X	1	1	1
Forest	1	X	3	6
Field	1	3	X	8,10
Road	1	6	8,10	X

Table 2. Multi-Spectral Channels which are optimal for the detection of region boundaries between four macro-level classes. The optimal channel(s) for the detection of a boundary between a class of the row dimension and a class of the column dimension is found as the table entry at the intersection of the row and column.

(ETL).[18] The ETL data are obtained from field measurements at the Martin Marietta ALV Test Site with the ETL spectroradiometer. They provide precise measurements of the spectral behavior of several terrain categories in the form of graphs of radiance and of reflectance with respect to a Halon Reference Standard, as a function of wavelength. These measurements may be used to extrapolate the average spectral reflectivities of the regions of interest. These estimates are then used to identify the Multi-Spectral Scanner channels which will exhibit the greatest discontinuity between regions for all possible pairings of the four classes. As a result of this analysis, a best discriminator set for the four classes was defined. It is presented in Table 2. Table 2. defines the best Multi-Spectral Scanner channel or channels that may be utilized for terrain region boundary detection. The optimal channel for detection of the region boundary between a specific terrain category along the row dimension and a terrain category along the column dimension is found at the intersection of the row and the column.

The parameters of the Texture Boundary Locator algorithm are set for each image according to the nominal range of the terrain class boundaries to be detected for each. For instance, channel 8 was identified as the best discriminator between the forest and field classes. Because fields are usually located within 5 to 100 yards of the ALV (as a function of the ALV's current mission profile) the forest-field region boundary will be wide. Therefore, the parameters of the Texture Boundary Locator algorithm used for this image are K = 9, N = 11. The values of the parameters N and K used for the calculation of each "plan" image for the experimental results presented in this paper are shown in Figure 2.

The evidence for region boundaries obtained by processing each of the five images with the Texture Boundary Locator algorithm are combined to produce a single gradient image Because the locations of the image where each of the five gradient images attains its maximum value are essentially disjoint, the gradient images may be combined additively. Therefore, each gradient image provides maximum support for the existence of a true inter-class boundary for the region boundaries it is designed to detect and collaborative evidence for the remaining region boundaries.

The next step of stage 1, the Multi-Spectral Edge Linking Relaxation algorithm, employs an edge confidence image for joining together incomplete boundaries. For an edge confidence image, pixel values range from A_1 to A_N, where A_N signifies the highest confidence that a pixel is an edge element. The confidence image locations with the intensity value A_N are those locations of the gradient image whose intensities were equal to or greater than the strictest threshold value for that image. For Multi-spectral Scanner imagery, three thresholds at the upper 15, 25 and 35 percent of the confidence image intensity levels were found to be sufficient to define the true region boundaries. The use of a greater number of thresholds did not improve the quality of the detected boundaries significantly.

The three steps of this algorithm are:

1. Label the maximum intensity pixels of the confidence image as edge elements.
2. Identify incomplete region boundaries. If no incomplete boundaries are found, stop; otherwise go to Step 3.
3. Link the incomplete boundaries found for Step 2. based on local edge evidence and the smoothness of the boundary. Go to Step 2.

The criteria employed for selecting new edge locations for Step 3, in order of precedence, are the following. 1. The edge location of maximum edge evidence of the set of neighbors of the endpoint, 2. The edge location of maximum edge evidence of the set of neighbors of the endpoint that causes the smallest change in the *curvature* of the incomplete boundary. Because new edge pixels are appended to incomplete boundaries until they become complete boundaries, the region boundaries defined by this algorithm are closed. After edge-thinning, statistics are calculated for each region, and these statistics are archived in a database for use by the knowledge-based region labeling algorithm.

2.3.2 Stage 2. Knowledge-Based Classification, Macro-Level - The rules of the knowledge-based region labeling algorithm are designed to locate regions of the plan image whose features are good matches to known features of five classes of terrain. The features of the five classes are archived in symbolic form in the knowledge base. The matching cri-

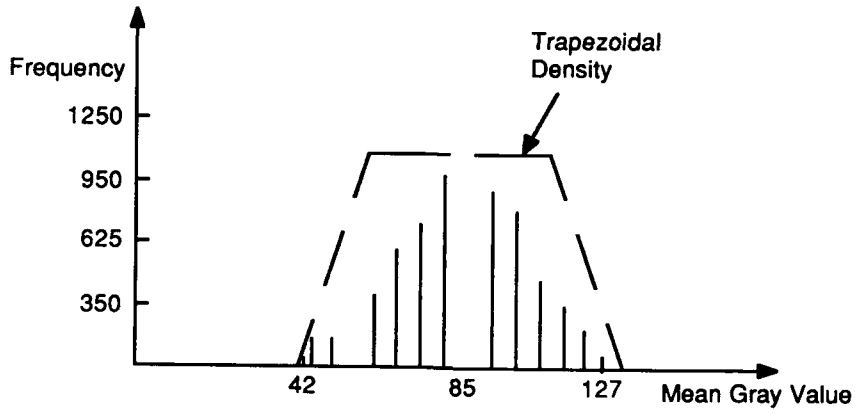


Figure 4. Histogram of 5x5 window means for *Forest* region in multi-spectral image 3.

terion will be described in the next paragraph. The five classes of terrain are:

$$\Theta = \{ \text{ sky, forest, mountain, field, road } \} \qquad (2)$$

The features used by the region labeling scheme are:

1. Spectral features, the mean and standard deviation of each region.
2. Locational features, the correlation of the location of the region on the image with the expected location as a function of the imaging system's orientation.
3. Relational features, a set of valid adjacency rules for neighboring regions.

The matching criterion employed is a pseudo-Bayesian measure for the conditional probability that the region observed is a member of a specific class, given the features of the region. The labels of the five classes are: c_i; i=1,...,5 for the classes sky, forest, mountain, field and road, respectively. The features employed to classify regions to one of the five classes are: f_{ij}; j=1,..., J_i, for class c_i; i=1,...,5. If the a priori probabilities for each class of Θ, $P(c_i)$; i=1,...,5, are known, then the conditional likelihood that a region is an instance of class c_i is:

$$p(c_i \mid f_{i1}, f_{i2}, \cdots, f_{iJ_i}) = \frac{P(c_i)\, p(f_{i1}, f_{i2}, \cdots, f_{iJ_i} \mid c_i)}{\sum_{k=1}^{5} P(c_k)\, p(f_{k1}, f_{k2}, \cdots, f_{kJ_k} \mid c_k)} \qquad (3)$$

where $p(f_{i1}, f_{i2}, \cdots, f_{iJ_i} \mid c_i)$ is the probability density of the features f_{ij}; j=1,..., J_i, conditioned on the fact that they are observations of class c_i.

Because of the immense variability of outdoor imagery, the features f_{iJ_i}, for each i, of a specific region are asymptotically independent as the size of the region becomes large. "Plan" image regions are large regions and, therefore, their features can be viewed as being independent. The previous expression may be written as:

$$p(c_i \mid f_{i1}, f_{i2}, \cdots, f_{iJ_i}) = \frac{P(c_i) \prod_{j=1}^{J_i} p(f_{ij} \mid c_i)}{\sum_{k=1}^{5} P(c_k) \prod_{j=1}^{J_k} p(f_{kj} \mid c_k)} \qquad (4)$$

The initial probabilities of the classes, $P(c_i)$; i=1,...,5, are all equal to 1 for our system. Equation (4) may be used to calculate the likelihood that a specific region is actually an observation of a specific terrain class.

The terms $p(f_{ij} \mid c_i)$; j=1,..., J_i; i=1,...,5 in equation (4) are very difficult to estimate for an arbitrary multi-spectral image. To obtain meaningful results with a Bayesian approach, it is necessary to approximate the density functions $p(f_{ij} \mid c_i)$; j=1,..., J_i; i=1,...,5 as uniform likelihood functions for an appropriate range of feature values.[6] These functions are often approximated as trapezoidal densities. An example of this technique is the density estimated for the 5x5 window means of multi-spectral channel 3, for the class forest, which is shown superimposed on the actual measurements of the feature in figure 4.

Each of the three categories of features for this stage; spectral, locational and relational, can be parameterized as conditional likelihoods. In a strict probabilistic sense, the conditional probabilities of relational features should depend on the assigned classes of both regions that effect the relation. That is, the likelihood should be a function of the form:

$$p(f_{ij} \mid c_i, c_k) \qquad (5)$$

where c_i is the class of the region being classified, and c_k is the class of the neighbor region.

Thus, Bayes Rule, as expressed in equation (4), does not apply for this feature and an alternative representation for the effect of the relational feature on the conditional class likelihood for class c_i is necessary. This technical difficulty is avoided for the current implementation of the HSGM algorithm by doing the following: The relational rules of the knowledge base are only applicable to adjacent

pairs of regions for which one of the regions has been classified with certainty to one of the five classes. When a region is classified with certainty for our system, all spectral and locational rules and at least one relational rule that effect the classification of the region have been evaluated, and the class label is that class that has the greatest conditional likelihood as calculated with equation (4).

Because relational features have to be evaluated for every region in the image, there must be an initial starting point where there is at least one region classified with certainty. The terrain interpretation algorithm uses a heuristic to solve this problem. All regions that qualify as a sky region with spectral and locational constraints alone are labeled to the class *sky with certainty*. Then the remaining regions are labeled with the full set of rules. When every region has been labeled, the algorithm stops.

2.3.3 Stage 3. Macro-Level Region Segmentation - The third stage of the terrain interpretation algorithm is the segmentation of each "plan" image region into subregions with the Texture-Gradient Edge Linking scheme of stage 1. The only differences between the implementation of this scheme for this stage and the implementation for the first stage are: 1. The group of images employed and 2. The coefficients that define the sensitivity of the Texture Boundary Locator algorithm. The set of images used for subregion boundary detection for each of the five terrain classes are those that exhibit sharp discontinuities between subregions of interest. The coefficients of the Texture Boundary Locator algorithm are the values that correspond to the range of the "plan" image region.

2.3.4 Stage 4. Knowledge-Based Classification, Micro-Level - The fourth stage of the algorithm is knowledge-based subregion labeling. The region labeling scheme is identical to the region labeling scheme of the second stage, except that limited relational features are used. The categories of spectral features are the same as for stage 2, with one enhanced feature; the ratio of subregion mean intensities for pairs of multi-spectral images.

2.3.5 Stage 5. Conflict Resolution - The last stage of the terrain interpretation algorithm is the resolution of conflicts. The purpose of this step is to transfer subregions from one "plan" image region to a neighboring "plan" image region if the classification of the subregion, obtained from stage 4, is in conflict with the parent region's classification. This step is necessary because "plan" image region boundaries are frequently offset from the true boundary location by as much as 10 pixels, due to the unsharp qualities of edges in outdoor scenes. This means that subregions that lie at the border between macro-level regions may be assigned to the wrong region. This step detects all subregions which may have been misclassified because they are adjacent to an incorrect "plan" image region boundary, and merges them into the neighboring region if there is no conflict with the class label of the neighbor. If the subregion cannot be classified as an element of either "plan" image region, it is labeled as an element of the class *unknown*. This is a useful technique for detecting regions for which the HSGM algorithm has no a priori information. Shadows may also be detected and properly classified by using physical world constraints.

3. EXPERIMENTAL RESULTS

The terrain interpretation algorithm has been implemented for stages 1-3. Work at implementing stages 4 and 5 is on-going. "Plan" image region classification results are good.

The HSGM algorithm is implemented on two computers at the Image Research Laboratory at the Honeywell Systems and Research Center. The feature detection stages, stages 1 and 3, are implemented in the C programming language on a VAX 11/750 computer under the Berkeley UNIX 4.3 operating system. The knowledge-based region labeling formalism, stages 2, 4 and 5, are implemented in a pattern-based reasoning language package, written in Common Lisp, on a Symbolics 3670. Data is transferred from one computer to the other by means of a ChaosNet File Transfer Protocol (FTP).

The performance of the first stage of the HSGM algorithm was evaluated for several Multi-Spectral Scanner images from the Collage 1 database[17] which was acquired at the Martin-Marietta ALV Test Area. A few of the experimental results obtained are described in the following examples.

Example 1. Figure 5 displays the 12 Multi-Spectral Scanner images of the image MULTI12 ("Segment O") from the Collage 1 data set. The 12 multi-spectral images are shown sequentially going from left to right across the page, where the first row begins at the top left corner of the figure, the next row is the one below it, etc. As a result of a scanner electronics malfunction, no data is available for channel 9 for any multi-spectral image of Collage 1, which is why channel 9 in figure 5 (3'rd row, 1'st column) is missing. Figure 6 is the luminance image (the Y image of the NTSC television standard for color imagery transmission) of the multi-spectral image MULTI12 over which the "plan" image region boundaries are superimposed. The segmentation of all terrain classes: sky, forest, field and road, are good. Note that the foothills are accurately segmented and that the occluding borders of hills in the foreground are detected as well. For the purpose of visual verification of the results, the luminance image without superimposed "plan" image region boundaries is presented in figure 7.

Example 2. Figure 7 is the luminance image of the multi-spectral image MULTI36 ("Segment V") from the Collage 1 data set, over which the "plan" image region boundaries are superimposed. All major region boundaries of the image have been detected, except for the left and right forks of the road. These region boundaries were not detected because they are located at too great a distance from the ERIM sensor. The range of distances at which road boundaries may be detected is a function of the parameters of the Texture Boundary Locator algorithm for multi-spectral images 8 and 10, which are the best channels for detecting road boundaries (see Table 2). Because the edges of road boundaries for these channels are not sharp, the coefficients of the Texture Boundary Locator algorithm are set for wide boundaries (K=9, N=11 for multi-spectral channel 8 and K=11, N=13 for multi-spectral channel 10). When road regions are

Figure 5. Display of 12 channels of Multi-Spectral Scanner image: MULTI12.

Figure 7. Luminance image: MULTI12.

Figure 9. Luminance image: MULTI36.

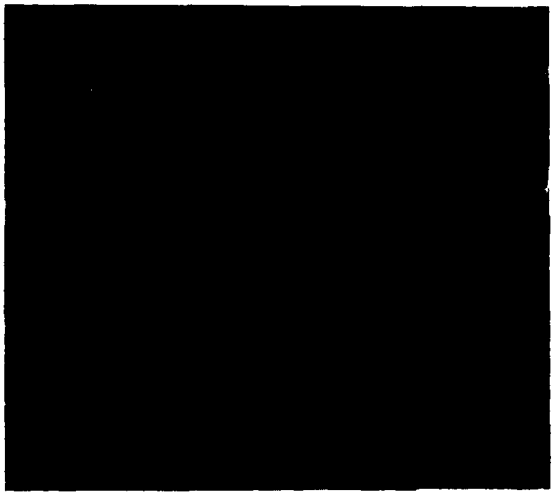


Figure 6. "Plan" image region boundaries: MULTI12.

Figure 8. "Plan" image region boundaries: MULTI36.

located at a great distance from the ERIM sensor, as the two forks of the road are, both borders of the road lie inside the texture gradient measurement window (see Figure 3). Therefore, the texture in their vicinity will appear to be homogeneous and the Texture Boundary Locator will not detect their boundaries. However, the Scene Model/Image Processing Requirements for vehicle guidance at a velocity of 10 km/hour, defined by Martin Marietta,[19] are 15 meters for the ERIM Multi-Spectral Scanner. Because the distance to the fork in the road is approximately 30 meters for MULTI36, the Scene Model that the HSGM algorithm defines is more than adequate for road following. With further improvements, such as a knowledge-base enhanced with elevation map, land cover map or range sensor information and the ability to incorporate temporal evidence[20] into the region classification procedure, it may prove to be satisfactory for cross-country navigation. For the purpose of visual verification of the results, the luminance image without superimposed "plan" image region boundaries is presented in figure 9.

4. CONCLUSIONS

A robust algorithm for the detection of structural region boundaries for terrain images was described. This is a novel technique for an application domain that traditionally has been approached with statistical methods. The problem of region classification is solved with knowledge-based techniques because of their efficiency for the task of processing symbolic feature data. Regions of the segmented multispectral image are labeled with an evidential reasoning approach because it counteracts the effects of incomplete or inaccurate knowledge in the knowledge base. Experimental results for macro-level region boundary estimation are described. The algorithm's current state of development is the following: the first three of five stages are implemented and stages 4 and 5 are now being evaluated and optimized. The initial experimentation with the algorithm has permitted us to acquire an appreciation for its potential, and on the basis of its strong performance for a very difficult image segmentation problem, we plan to continue the development of this algorithm.

REFERENCES

References

1. D. M. McKeown, Jr., "Knowledge Based Aerial Photo Interpretation," *Photogrammetria* **39** pp. 91-123 (1984).
2. M. Nagao and T. Matsuyama, *A Structural Analysis of Complex Aerial Photographs*, Plenum Press (1980).
3. W. A. Perkins, T. J. Laffey, and T. A. Nguyen, "Rule-based interpreting of aerial photographs using the Lockheed Expert System," *Optical Engineering* **25**(3) pp. 356-362 (March 1986).
4. L. Sauer and J. Taskett, *Cultural Feature and Syntax Analysis for Automatic Acquisition*, SPIE Conference on Processing of Images and Data from Optical Sensors (1981).
5. M. D. Levine and A. M. Nazif, "Low Level Image Segmentation: An Expert System," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **PAMI-6**(5) pp. 555-577 (September 1985).
6. Y. Ohta, *Knowledge-based Interpretation of Outdoor Natural Scenes*, Pitman Publishing, Inc. (1985).
7. S. M. Rubin, "Natural Scene Recognition Using Locus Search," *Computer Graphics and Image Processing* **13** pp. 298-333 (1980).
8. D. A. Landgrabe, "Analysis Technology for Land Remote Sensing," *Proceedings of the IEEE* **69**(5) pp. 628-642 (May 1981).
9. P. H. Swain, "Advanced Interpretation Techniques for Earth Data Information Systems," *Proceedings of the IEEE* **73**(6) pp. 1031-1039 (June 1985).
10. J. A. Richards, D. A. Landgrebe, and P. H. Swain, "Pixel Labelling by Supervised Probabilistic Relaxation," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **PAMI-3** pp. 181-191 (1981).
11. C. A. Harlow, M. H. Trivedi, R. A. Conners, and D. Phillips, "Scene Analysis of High Resolution Aerial Scenes," *Optical Engineering* **25**(3) pp. 347-355 (March 1986).
12. A. Rosenfeld, C-Y. Wang, and A. Wu, "Multispectral Texture," *IEEE Transactions on Systems, Man and Cybernetics* **SMC-12**(1) pp. 79-84 (January/February 1982).
13. J. M. Brayer, P. H. Swain, and K. S. Fu, "Modeling Earth Resources with Satellite Data," in *Syntactic Pattern Recognition, Applications*, ed. K. S. Fu,Springer-Verlag (1977).
14. M. Goldberg, D. G. Goodenough, M. Alvo, and G. M. Karam, "A Hierarchical Expert System for Updating Forestry Maps with Landsat Data," *Proceedings of the IEEE* **73**(6) pp. 1054-1063 (June 1985).
15. C. A. Therrien, "An Estimation-Theoretic Approach to Terrain Image Segmentation," *Computer Graphics and Image Processing* **22** pp. 313-326 (1983).
16. H. Nasr, B. Bhanu, and S. Schaffer, *Guiding the Autonomous Land Vehicle Using Knowledge-Based Landmark Recognition*, Proceedings of the DARPA Image Understanding Workshop (Feb. 1987). (These Proceedings)
17. J. A. Allison, "COLLAGE: A Collection of Sensor Images for the ALV Test Area," Martin Marietta Corp. (December 1985).
18. J. N. Rinker, J. P. Henley, and M. B. Satterwhite, "Terrain Data Base - Air Photo Analysis, Martin Marietta ALV Test Site," Martin Marietta Report (January 1986).
19. J. Lowrie, "The Autonomous Land Vehicle Second Quarterly Report," Martin Marietta Corp. (September 1985).
20. B. Bhanu and W. Burger, *DRIVE - Dynamics Reasoning from Integrated Visual Evidence*, Proceedings of the DARPA Image Understanding Workshop (Feb. 1987). (These Proceedings)

DESIGN OF A PROTOTYPE INTERACTIVE CARTOGRAPHIC DISPLAY AND ANALYSIS ENVIRONMENT

Andrew J. Hanson, Alex P. Pentland, and Lynn H. Quam

SRI International (Artificial Intelligence Center)
333 Ravenswood Avenue, Menlo Park, California 94025

Abstract

We discuss design issues for an interactive scene modeling system appropriate for cartographic tasks. Among the major goals of such a system are the ability to enter and display cartographic features registered to geographic coordinate systems, to handle cartographic data base operations, and to support automated and semiautomated feature compilation. In addition, the system should be suitable for use as a softcopy, interactive cartographic display system that could replace hardcopy maps for an end user. A prototype for such a system has been implemented and is being actively developed. We present some of the features and capabilities of the existing system, and illustrate its use in several scenarios.

1 INTRODUCTION

Manual photointerpretation of images is a difficult and time-consuming step in the compilation of cartographic information. Organizations that are responsible for map generation are faced with increasing volumes of data to be processed, as well as with demands for an increasing variety of cartographic products. Manual photointerpretation techniques do not seem to be able to meet the projected requirements; thus it is hoped that these needs can eventually be met by using digital image data and automated image-understanding approaches.

Unfortunately, it is extremely unlikely that complete automation can be achieved with current image-understanding techniques. Among the concepts that might be used to circumvent this problem in the near term, and perhaps even in the long term, are the following:

- **Computer-Assisted Enhancement of Human Productivity.** We can exploit computer-automated tools for enhancing the productivity and capabilities of the individual human photointerpreter. It is reasonable to expect that a division of labor can be defined in which decisions requiring substantial expertise and knowledge of the context are carried out by the human, while tedious tasks requiring less abstract knowledge can be sent to the computer. The human in such a scenario plays the role of a supervisor who directs the attention of the automated processes, then evaluates and edits the results to conform with his or her superior knowledge of the task. To accomplish the desired productivity enhancement, a well-engineered human interface is absolutely essential.

- **Conversion to Softcopy Cartographic Viewing Systems Customizable by the End User.** Hardcopy cartographic products are typically designed for particular task domains. Attempts to economize by making cartographic products that cover many different needs are counterproductive because extracting task-specific information becomes too difficult for users. By providing computer data bases with flexible mechanisms for selecting the information to be displayed, we can permit the user to create a map specifically adapted for the task at hand, while retaining the freedom to change tasks quickly within the same environment. Furthermore, since computer methods may be used to construct customized data displays, to update rapidly changing cartographic information, and even to simulate natural scene views in real time, the overall effectiveness and utility of softcopy viewing methods could in many cases surpass those of hardcopy cartographic systems.

Research on computer-based system designs that can simultaneously enhance the capabilities of an individual human analyst and adapt to the cartographic requirements of an end user is therefore of great interest.

In this paper we discuss the design and implementation of a prototype system with the desired capabilities. Major portions of the system have been implemented, tested, and rewritten several times, so that our concepts of the design issues have been significantly seasoned by practical experience with the prototype. Section 2 gives an overview of the design objectives and the way they have been met in the current system. Section 3 presents several scenarios showing how the system can be used; these examples illustrate some techniques that are unique to the interactive computer-based analysis environment. We conclude with a discussion of our plans for future work.

2 SYSTEM DESIGN

We believe that the following specific capabilities are among those needed in an interactive cartographic display and analysis environment that meets the needs outlined above:

1. Support creation, editing, interactive manipulation, and realistic rendering of objects making up arbitrarily complex three-dimensional (3-D) scenes.

This research was supported principally by the Defense Advanced Research Projects Agency under Contract No. MDA903-86-C-0084; some aspects of the research were also supported by National Science Foundation Grant Nos. DCR-8312766 and IST-8511751.

2. Allow such objects to be merged with geographically precise data bases of images, terrain models, and maps.

3. Handle multiple overlapping views of the same geographic area, including but not limited to stereographic image pairs, while maintaining the correct geographic locations of all interactively-moving objects in each view.

4. Generate synthetic scene views using object data bases combined with terrain models, images, and feature modeling knowledge.

5. Incorporate data bases of semantic object relationships and domain knowledge, along with the ability to answer interactive set-membership queries concerning data base contents.

6. Support automated, semiautomated, and manual cartographic compilation procedures within the system.

7. Support the incorporation of knowledge-based systems that correctly deal with geographic and temporal relationships as well as with the context of cartographic compilation.

The major characteristics of the current system are summarized in Figure 1. In broad overview, the system works on a body of *information sources* and provides the user an elaborate set of *interpretive tools* with which to manipulate and edit cartographic information displays.

2.1 Information Sources

Among the types of information that should be handled by the system are the following:

- **Images.** Digitized photographs or similar sensor data from geographic areas relevant to modeling and feature extraction tasks. Associated with the images should be supplementary data such as geographic coordinates, camera models, illumination information, exact times that allow reconstruction of sun position for outdoor scenes, sensor characteristics, sensor response information, and atmospheric properties at the time the images were generated.

- **Digital Terrain Elevation Data.** When available, terrain elevation data should be registered to the corresponding images. This permits the construction of accurate synthetic images, as well as the generation of cartographic products containing elevation information.

- **Compiled Feature Data.** Cartographic feature data bases are needed for the generation of cartographic products meeting the needs of tasks for which an image alone is insufficient. Ideally, the system should support the interpretation of precompiled feature data bases, the editing of any item in such a data base, and the interactive entry of new data base features.

- **Feature Models.** In order to support the entry of new features, a library of predefined prototype feature models is required. Such models would be used either by an analyst generating a distributable feature data base product, or by an end user updating an existing data base. Feature models typically interact with utilities that allow them to be instantiated, moved to correct geographic positions, and adjusted with respect to their internal parameters.

2.2 Interpretive Tools

The interactive needs of the user are met by several subsystems of software tools oriented toward cartographic compilation tasks. The interpretive tools currently available or planned in the immediate future include:

- **ImagCalc** *(TM).* The SRI ImagCalc system provides a comprehensive interactive environment for viewing, examining, and analyzing digitized imagery. The interactive environment itself is implemented using a complete library of programmer's tools that are available for developing new image processing application programs and systems. ImagCalc supports the use of multiple concurrent image-containing graphics windows, each of which is associated with a stack of image displays. Full facilities for generating plots and overlay graphics are available. ImagCalc also supplies mechanisms for accessing extremely large images directly from disk storage, without requiring that the image occupy the computer's possibly limited virtual memory.

- **SuperSketch** *(TM).* The SRI SuperSketch modeling system [see, for example, Pentland, 1986] contains complete facilities for creating general shapes based on superquadric algebraic forms and their deformations. Utilities are available for generating fractal textures and other realistic coloring and texturing in rendered scenes. Composite objects with negative components are available to permit the carving of complex holes in a shape. The methods used by SuperSketch are particularly useful for the construction of natural and irregular objects. Object models generated by SuperSketch can be passed directly to the cartographic modeling system and built into cartographic scenes.

- **Cartographic Modeling Tools.** To meet some of the particular needs of cartography, the system contains an extensive cartographic modeling facility. Typical models include 3-D cross-hairs, 3-D labels, arbitrary planar-faced objects, smoothly shaded objects represented by triangular grids, digital terrain models, closed-curve area delineations, linear features, and ribbon-like features. Various types of buildings and cultural structures are represented as planar-faced objects, while SuperSketch models are represented within this context as a type of smoothly shaded object. Among the facilities available are:

  - Interactive adjustment of 3-D model parameters.
  - Precise manual entry of the cartographic coordinates of an object.
  - Arbitrary adjustment of the camera model.
  - On-demand adjustment of object altitude to match the elevation in the digital terrain model.
  - Matching of object altitude to the disparity in a stereo pair.
  - Display of the path of a ray of sunlight through any object vertex terminating at the terrain model surface.

  Many different viewpoints of a particular scene's object models may be maintained and kept in view on different windows of the screen. Motion of an object can be displayed simultaneously in multiple views so 3-D alignment is achievable with or without a stereoscopic display. Among the different motion modes available is the capability of moving

an object along the camera ray of a particular image, so that in other views the object moves on the epipoles of each view. Several different digitized images can be displayed with the same geographically positioned cartographic objects; this type of interimage alignment is almost trivial in this system, but can be difficult to achieve using manual techniques.

- **Synthetic Scene Generation.** When a digital terrain elevation model is available, any image may be used to generate a synthetic view of the scene from an arbitrary camera position. This facility computes a view of the elevation model with a resolution-dependent resampling of the image projected onto it [see, for example, Quam, 1985]. Modeled objects can be incorporated into such scenes either as wireframe overlays or as simulated grey-scale renderings utilizing the lighting model. One can construct simulated monoscopic or stereoscopic movies that greatly enhance the user's perceptions of the scene by chaining together a sequence of such views and displaying them in rapid succession. For certain tasks, such a simulated movie of the terrain appearance can be vastly more useful than a conventional map.

- **Cartographic Data Base Facilities.** Several coordinated efforts are now underway at SRI to design data bases appropriate for cartography and navigation. While these particular capabilities have not yet been incorporated directly into the system, we have implemented elementary data base operations that allow the selection and grouping of cartographic features. As the data base design work matures, we expect to add the new capabilities to the current prototype. We will then be able to handle data base construction and interactive queries interrogating the cartographic data structures.

- **Semiautomated Application Facilities.** The capabilities and utilities of the existing system provide a very flexible framework in which to build applications. With our current interactive facilities, it is natural to emphasize application programs that have a substantial interactive component. We envision the system starting out with the manual interfaces required for many cartographic compilation tasks, then progressively acquiring the subsystems needed to automate various subtasks, and finally being able to achieve some types of work in a completely automated fashion.

We believe that providing an environment in which to explore the evolution from manual computer-based cartography to automated cartography is one of the most important features of the effort described here. With existing technology, certain tasks, such as those depending on extensive background knowledge, common sense, and reasoning, can be carried out by humans much more effectively than by any computer system; similarly, for many tasks, such as those involving extensive computation and repetition of well-understood algorithms, computer automation is much more efficient than a human could be. Finally, there is a middle ground, where a complex operation can be carried out most efficiently using techniques that involve close cooperation between a human operator and an automated computer. The challenge, then, is to discover those problem domains and techniques that allow humans and computers to cooperate and produce results that exceed the capabilities of either functioning alone.

3 ILLUSTRATIVE SCENARIOS

In this section, we describe several actual scenarios that can be carried out in the current implementation of this system.

3.1 Object Construction by Shadow Alignment

For our first task, let us see how an accurate 3-D building model would be constructed by exploiting various features of the system. In Figure 2a, we show an image containing tall buildings; supplementary data for this image include the elevation model, solar illumination geometry, and a camera model. First, we create a default rectangular building model and use a utility to drop one corner to the ground level as indicated by the elevation model. Next, we interactively adjust the length and width of the building model until they coincide with the appearance of the building roof in the image. We now adjust the height of the building. Using another utility to drop a sun ray from the building corner to the ground (as estimated from the elevation model), we see the display shown in Figure 2b; by adjusting the height until this ray coincides with the building shadow, we can obtain a relatively accurate 3-D model of the building. An experienced user can accomplish this entire procedure in 5 or 10 seconds.

A different approach is available if we make use of the system's scene simulation capabilities. Using the solar illumination geometry, we can construct a camera model that corresponds to an orthographic projection of the scene through the location of the sun itself. In this view, motion along any sun ray leaves a point in the synthetic image invariant. Similarly, all shadows of building corners must appear exactly aligned with the building corners themselves, to within the accuracy limits of the elevation model. Using this "solar" camera model, we construct the synthetic scene view shown in Figure 2c, and align the building corners with the shadows directly in the synthetic image. In Figure 2d, we show another synthetic view containing a rendering of several 3-D building models constructed using these techniques.

3.2 Stereographic Road Delineation

Suppose now that we have a stereo pair of images or a distinct pair with different camera parameters, while our task is to enter a curve that follows a road in the image. Starting from the image pair in Figures 3a and 3b, we begin laying down the points of a curve following the road. At each point, we can check the appearance of the curve in both pairs of images simultaneously. In particular, it is very effective to use the utility that fixes a point of the curve to lie on the camera ray in one image while moving it along an epipole in the stereo view. We construct and check each node of our curve in this way, and then display the pair using a 3-D spline fit, as shown in Figures 3c and 3d.

3.3 Hand Segmentation

Finally, suppose our task is to create a manual segmentation of a particular area. We begin with the image in Figure 4a, and create a curve. By interactively inserting, deleting, and moving nodes of the curve, we construct the delineation shown in Figure 4b, and its spline fit shown in Figure 4c. This is a typical situation for which, once a human operator has provided a good starting point, an automated process can be invoked to

refine the accuracy of the result. Here, for example, we use a gradient-ascent utility developed by Leclerc and Fua [1987] to find the best local boundary with strong edges that is near the initial curve; the result of this automated refinement operation is shown in Figure 4d.

4 PLANS FOR FUTURE WORK

The current implementation of this system has many capabilities that are of interest for manual and semiautomated digital cartography. However, much remains to be done to explore and evaluate the effectiveness of the techniques that might be used. Among the specific efforts that we plan to undertake in the near future are the following:

- **Provide Symbolic Access to Scene Objects.** Any cartographic object can be viewed either as a geographical entity, with a display mode that is typically registered to the image in which it appears, or as a symbolic entity, with relationships to other features in the cartographic context. Objects that are composites or that are components of composite objects are sometimes best manipulated in terms of their logical identity, rather than their geographic one. To meet this need, we plan to develop a dual-mode access system that allows the user to construct symbolic graphs corresponding to geographic feature clusters, to display symbolic graphs, and to interact with such graphs in a way that displays the geographic meaning of the graph nodes simultaneously with the semantic meaning. To accomplish this, we will probably incorporate other SRI work on symbolic cartographic data bases and graphical knowledge representation methods into our cartographic interpretation capabilities.

- **Extend Scene Simulation Capabilities.** The current system supports only relatively simple rendering and surface characteristics. One of our next goals will be to add the ability to use digital image information from multiple images in scene simulations. We will also add prestored texture models for certain types of objects so that we can simulate additional details of objects with internal structure. Finally, we plan to add the capability of storing the characteristics of features that are significant for SAR image simulation, so that we can ultimately generate synthetic SAR images as well as optical images. We also intend to investigate the problem of simulating scenes at times of day different from those of any available source image. This is technically very difficult to do correctly when texture mapping from real images, but some reasonable approximation techniques might be found that would be very helpful for human viewers.

- **Support Semiautomated Feature Extraction.** One of the long-term goals of the system is to provide support for a wide range of manual, semiautomated, and automated cartographic compilation procedures. In the short term, our first goal will be to merge some current SRI work on feature extraction into the cartographic modeling system. For example, the building outlines produced by the automated feature extraction system of Fua and Hanson [1986, 1987] can be used as initial building-top outlines in the modeling system; with such a starting point, it is straightforward to determine the 3-D extent of the building using the shadow-driven techniques described above.

- **Exploration of User Interface Design.** Among its other capabilities, the system we have developed is ideal for testing out the concept that computer displays can eventually replace *both* manual photometric compilation and the use of hardcopy maps. In particular, new ideas concerned with the use of digital data bases for the end-user assembly and soft-copy viewing of cartographic products can be prototyped and tested in this environment. Our last major objective is therefore to continue to explore many different concepts for the user interfaces of such systems.

REFERENCE

P. Fua and A.J. Hanson, "Resegmentation Using Generic Shape: Locating General Cultural Objects," Pattern Recognition Letters (1986) *in press*; "Using Generic Geometric Models for Intelligent Shape Extraction," in this Proceedings (1987).

Y. Leclerc and P. Fua "Finding Object Boundaries Using Guided Gradient Ascent," in this Proceedings (1987).

A.P. Pentland, "Perceptual Organization and the Representation of Natural Form," Artificial Intelligence **28**, pp. 293–331 (1986).

L.H. Quam, "The Terrain-Calc System," in Proceedings of the Image Understanding Workshop (1985).

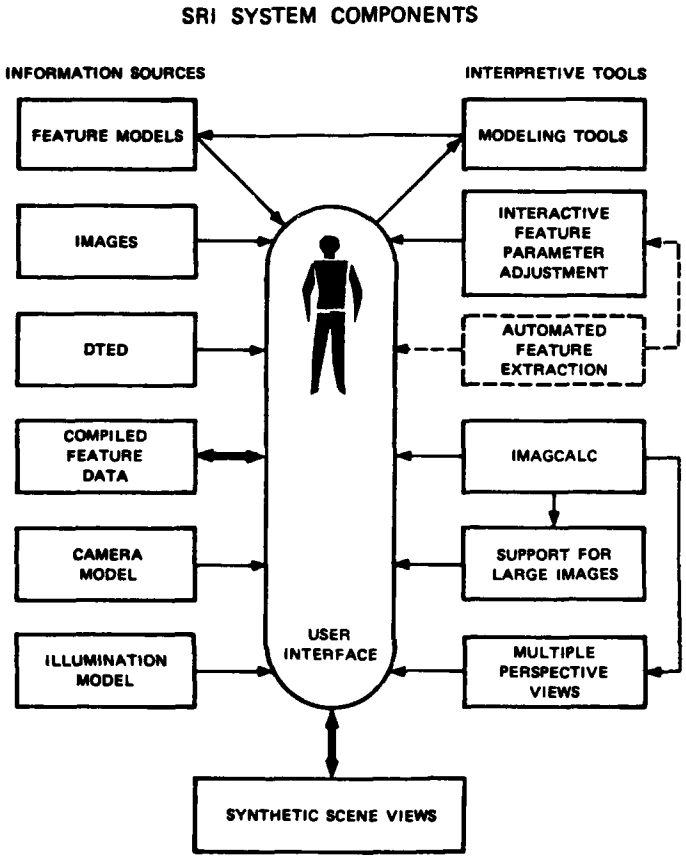


Figure 1: Block diagram of the major components of the SRI prototype cartographic analysis and display system.

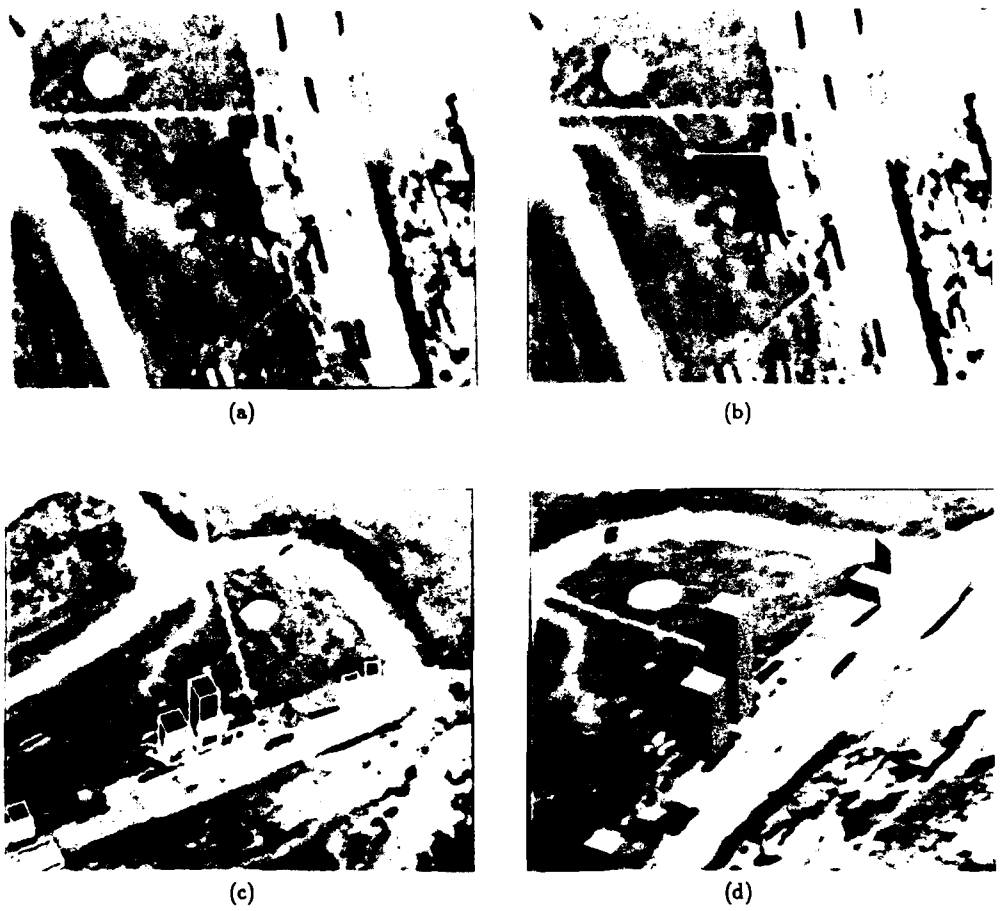


Figure 2: (a) An image containing tall buildings with distinct shadows. (b) The sun ray from the building corner being adjusted to the ground. (c) The building model viewed against its shadow in a synthetic image taken from the viewpoint of the sun itself. (d) Another synthetic view of the building model, this time rendered into the scene using a simple Lambertian reflectance model.

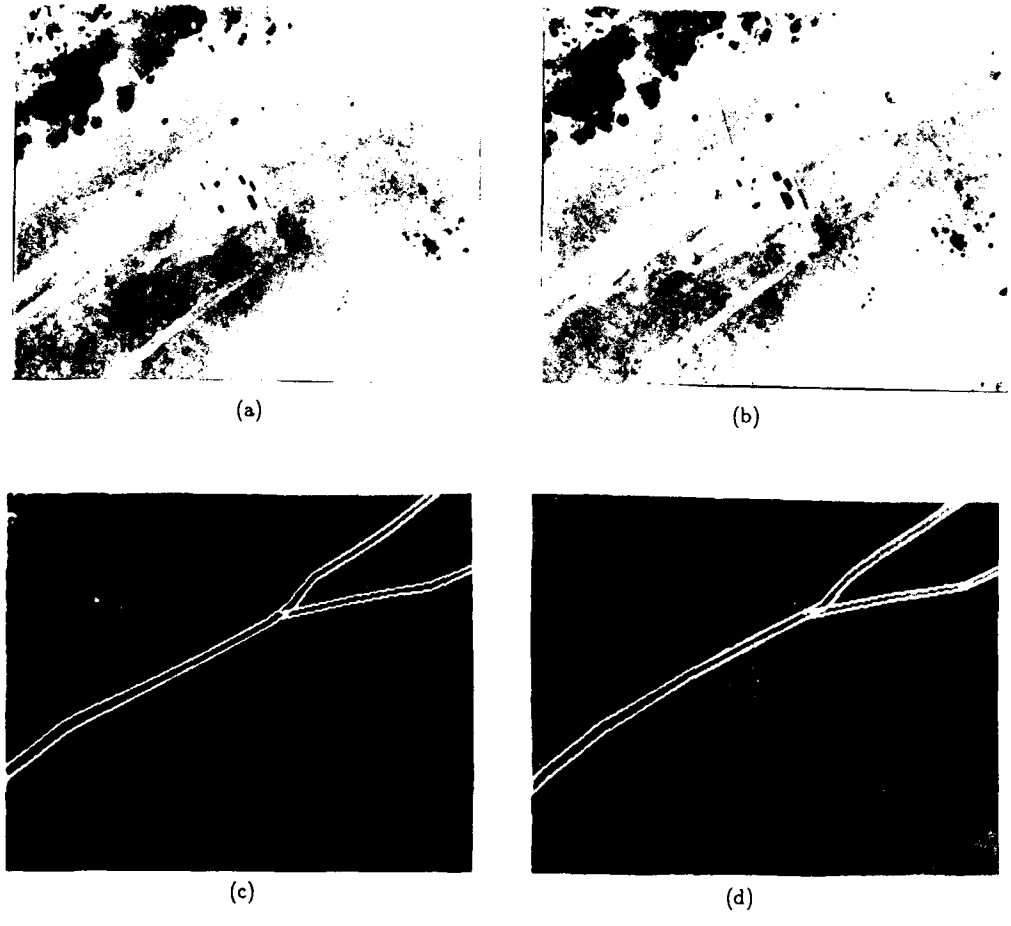


Figure 3: (a) Right stereoscopic view of terrain containing a road. (b) Left stereoscopic view of terrain containing a road. (c) Right view of the stereographic curve constructed to follow the road. (d) The corresponding left view of the curve.

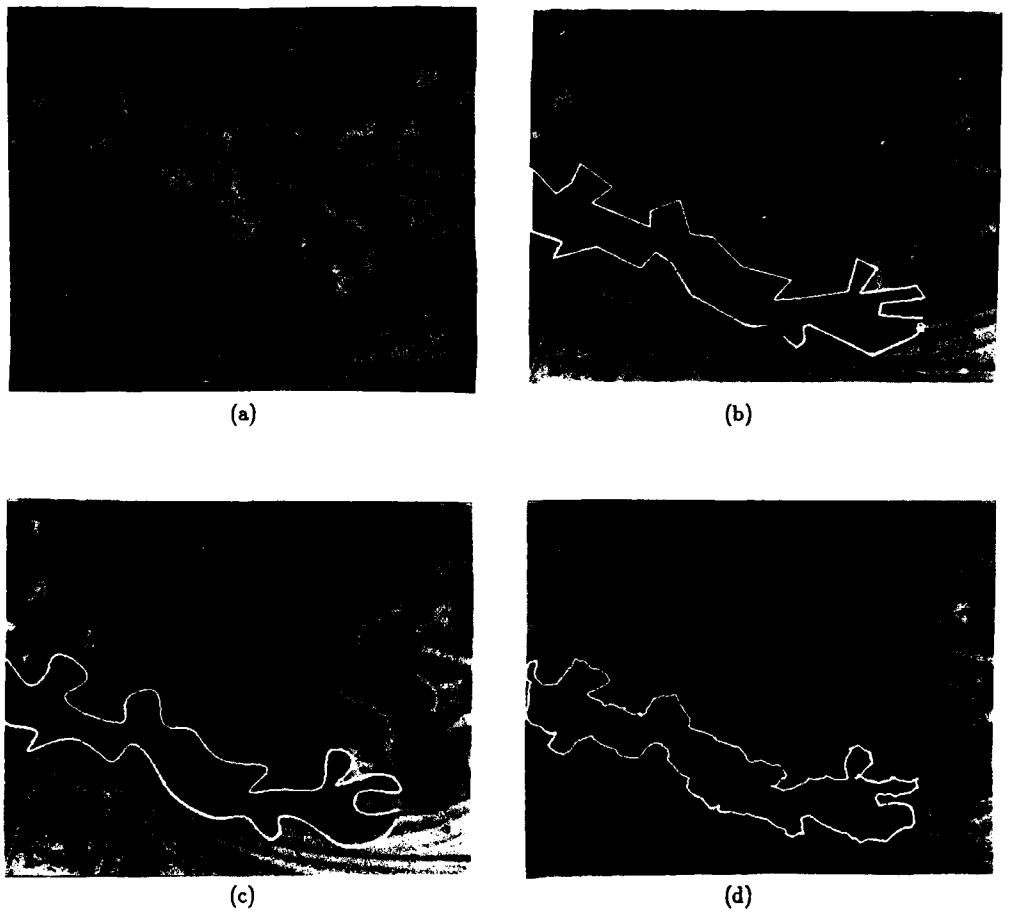


Figure 4: (a) An image containing a region to be segmented manually. (b) The raw set of straight line segments generated during the manual editing of the curve outlining the region. (c) A 3-D spline fit to the curve. (d) A region delineation computed using a gradient ascent technique to find the best boundary with a strong edge that is near to the manually-entered boundary.

The Image Understanding Architecture

Charles C. Weems, Steven P. Levitan*
Allen R. Hanson, Edward M. Riseman

Department of Computer and Information Science
University of Massachusetts
Amherst, MA 01003

ABSTRACT

This paper begins with the motivation and rationale for the design of the Image Understanding Architecture (IUA); a massively parallel system for supporting real-time image understanding tasks and research in computer vision. A small-scale demonstration prototype of the IUA is currently being constructed by the University of Massachusetts and Hughes Research Laboratories. The remainder of this paper presents an overview of the IUA and some details of the architecture. We conclude with a brief discussion of the IUA operating and software development environments.

1. INTRODUCTION

Machine vision is one of the most computationally intractable domains of artificial intelligence research. It requires that an interpretation of a changing scene be updated with every new video frame: once every thirtieth of a second. Each video frame contains three quarters of a million color-intensity data values which comprise the picture elements (pixels) of the image. Performing a single operation on each of these pixels requires executing about 23 million instructions per second just to keep up with the input. Of course, the computation needed to perform image interpretation is much more than one operation on every pixel in an image. Some researchers believe it could be as much as 6 or 7 orders of magnitude more computation.

An interpretation is a high-level description of the environment from which the image was taken. It must be in a form that is suitable for planning such diverse activities as robot arm and hand motion, obstacle avoidance by a vehicle, or aircraft navigation. Automatic scene interpretation requires the construction of at least a partial description of the original environment, rather than a description of the image itself. It involves not only labeling certain regions in an image, or locating a single object in the viewed scene, but often requires a three dimensional model of the surroundings, with associated identification in the image of the 2-dimensional projections of these 3-dimensional models. Figures 1 and 2 show a simple example of an interpretation of a house scene.

Figure 1. Raw Image

Interpretation is made even more difficult by the fact that the image data is inherently ambiguous and incomplete. Let us consider a very difficult example, a window viewed from the exterior of a house, in order to make our point. One must expect that the interior might be visible; through a portion of the window, another portion, partially superimposed on the first, might reflect part of the outdoor scene, and might contain several specular reflections; yet another part of the window could appear opaque; finally, the lower portion of the window could be obscured by shrubs. This example is a situation that is impossible to interpret with pattern recognition techniques, because the window does not have a unique pattern. It is a collage of visual patterns, many of which are only partially visible. While there has been a little success[Faugeras and Price, 1981] in applying a Bayesian classification viewpoint to some select subproblems in Computer Vision, there are many difficulties and we believe standard statistical ap-

* Current address: Department of Electrical Engineering, Benedum Hall, University of Pittsburgh, Pittsburgh, PA 15261

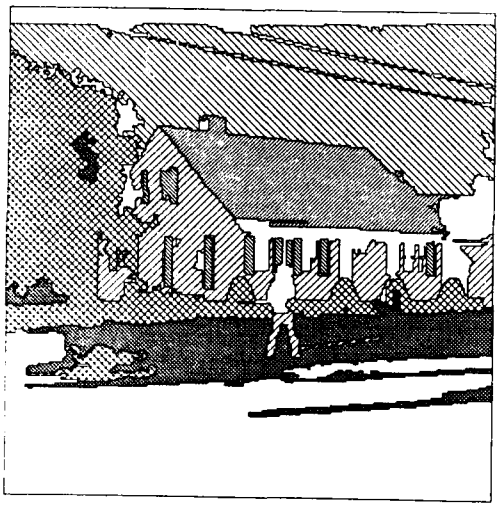


Figure 2a. Example Image Interpretation

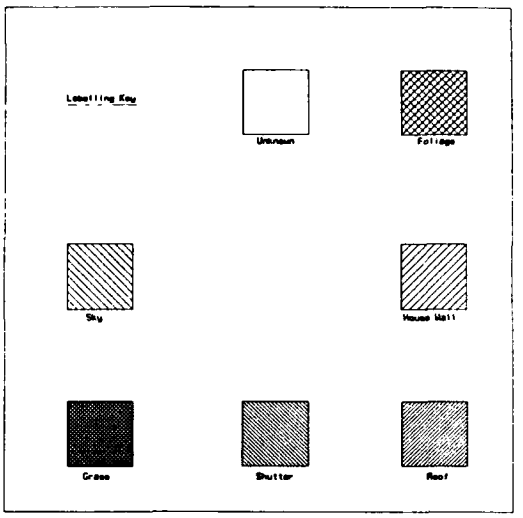


Figure 2b. Labeling Key

proaches generally suffer from insoluble problems. Classical pattern recognition techniques are not powerful enough by themselves to produce effective classifications in the domains we wish to consider.

While this example scene might be beyond the current capabilities of computer vision, it is clear that correct interpretation of the window image would require that we take into account the larger context of the overall scene, and knowledge about the world in general. Because the ambiguous data is part of a house, and is positioned where a window might appear, it might be possible to hypothesize that it is a window. Special processes may then be brought to bear to verify this hypothesis. If the hypothesis is confirmed, then additional knowledge may be used to fill in missing data and use weak cues to infer additional related objects consistent with this context. For example, the interpretation may state that the lower portion of the window, although obscured, is probably rectangular.

Thus, we are inferring the presence of a portion of the window from knowledge about houses and windows in general. Inference via stored knowledge and the reduction of ambiguity from "low-level" image analysis are a part of what is referred to as "high-level" vision processing. Without knowledge-based processing it would be impossible to interpret large portions of many images. The approach to knowledge-based vision that follows is derived from the VISIONS system development project for the analysis of natural scenes at the University of Massachusetts (UMass),[Hanson and Riseman, 1974, 1978b, 1986; Parma et al., 1980; Reynolds et al., 1984; Weymouth 1986].

For high level interpretation, the principal unit of information is a symbolic description of an object, or a set of image events, sometimes referred to as image features or symbolic tokens, extracted from the image. The description includes relationships both to other 2 dimensional (2D) symbolic tokens extracted from the sensory data, such as lines, regions, and surfaces and to other objects in the 3 dimensional scene being viewed. It also includes pointers to elements of general knowledge that have been used to support the interpretation process. At this level the representation involves semantic concepts of "object-classes", their sub-classes, their "part-decomposition" and the various types of relations between them.

Knowledge-based processing, however, can only take place after a certain amount of low-level processing has occurred. Low level processing principally involves classical image processing techniques such as contrast enhancement, and computer vision techniques of segmentation and feature extraction, edge detection, and region labeling. At the low level, the principal unit of information that is being processed is the pixel, or picture element, consisting of the color or intensity values of the image, and possibly range data for the visible surface element associated with each pixel.

There is no simple computational transformation that will map arrays of pixel values onto the stored symbolic concepts represented in the high level knowledge base. It has become generally accepted [Hanson and Riseman, 1980] that many levels of representation (data abstraction) and many stages of processing must take place to reliably interpret a scene. We will refer to the set of representations between the low and high level data structures as the intermediate representation.

The intermediate level of representation bridges the gap between the low and high levels. At the intermediate level the basic unit of information is a description of an image event extracted from the image data, and referred to as a token. Examples of token classes (or types) are lines; intensity, color or texture regions; and surfaces. Processing at the intermediate level may then involve grouping these

token events into more complicated structures such as rectangles, planes, or sets of parallel lines.

The intermediate representation is treated as a database by the high level, which queries it in order to form initial hypotheses about the content of the image. Following this, verification of hypothesis and resolution of conflicting hypotheses often requires the high level to initiate further processing in the low and intermediate levels in order to extract additional information from the image.

Image interpretation may thus be characterized as involving three different levels of processing, each with its own specific class of information. Additionally, those levels must be able to interact through bottom-up transfers of information (Figure 3) and top-down control of processing. The low, intermediate, and high levels of representation and processing, together with our understanding of how those levels interact, provides the basis for the design of our Image Understanding Architecture (IUA) presented in section 3.

2. PARALLEL ARCHITECTURES FOR KNOWLEDGE-BASED VISION

The motivation for building a "vision machine" or image understanding architecture stems from the need to support a set of complex operations on massive amounts of data at high speeds, and provide an environment for experimentation that justifies the enormous expense in software that is necessary to build a vision system. In this section we discuss the requirements for such an architecture and review some possible architectures and organizations.

2.1 Architectural Requirements for Vision

Some of the architectural issues to be addressed for vision come directly from the specific requirements of the problem:

* The ability to process both pixel and symbol data.
* A fast processing rate for huge amounts of sensory and intermediate level data.
* The ability to transform an image into a set of meaningful symbols that describe it.
* The ability to select particular subsets of data for varying types of processing.
* Feedback mechanisms that allow focusing of attention and data-directed processing, without having to dump the image to some "host" for external evaluation.

Beyond these requirements, two significant architectural implications can be derived from an understanding of our approach to the computer vision problem:

* Multiple levels of representation and stages of processing are essential and require very different types of processing elements.
* Fine grained and high speed communication and control is required both among the processes at each level and between the different processing levels.

We first discuss the computational requirements needed at different processing levels and then move on to discuss the the communication and control issues involved in vision processing.

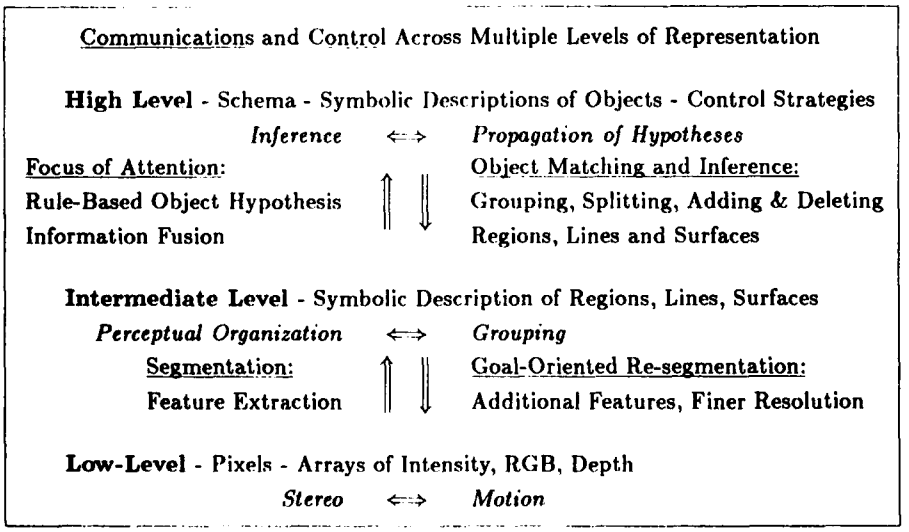


Figure 3. UMass VISIONS Image Interpretation System

2.2 Processing Characteristics for Multi-Level Architectures

We have mapped the three levels of abstraction discussed above into three levels of architectural requirements with each level having the appropriate architecture for the set of tasks at that level. The simplest level is the low-level where uniform computation is applied at local points, often at each pixel, across the image. At the low level we need to perform operations such as smoothing, edge detection, region segmentation, feature extraction, and feature matching between frames in motion and stereo processing. Here, each pixel or a neighborhood around each pixel, must be processed and there are large amounts of local processing to be applied in SIMD fashion. Some of these operations may also require processing over larger image distances.

In addition to high-speed low level operations, we need to be able to quickly load the data and to test the results of partial processing. To perform multiple iterations of low level operations under control of higher level processes requires fast, concise feedback from the low level processes to the intermediate and high level processes as well as data-dependent control of the low level processes by the higher ones.

At the intermediate level our concerns are for the symbolic tokens associated with extracted image events. The types of operations needed at this level are partitioning and merging, which transform these tokens into more useful structures in conjunction with high level hypotheses about the possible interpretations of events in the image.

The intermediate level also requires specialized processors. Consider the large numbers of line and region fragments that can be generated by even the most effective low level algorithms. To perform grouping operations (e.g., merging and splitting of regions, or linking and reorganizing lines) we need a large amount of "less local" communications. We need to match fragments of lines and merge them across large fractions of the image. Similarly, regions need to be merged and compared with others from possibly non-contiguous areas.

This data reduction through abstraction process is the key to intermediate level processing. For it to be done quickly requires architectural support for both inter-level and intra-level communication as well as a flexible data manipulation repertoire of instructions. Thus, the intermediate level must operate as a server for the queries made by the higher level processes, support data reduction processes, and provide control and evaluation mechanisms for the lower level processes.

At the high level we are concerned with semantic processing involving mechanisms for focus of attention, the formation and verification of object hypotheses, and knowledge based inference using complex control strategies from multiple knowledge sources. This type of processing involves extensive symbolic computation.

High level operations generate and test hypotheses based on available data provided by the low and intermediate levels of processing and request new data to be abstracted from the image if needed. Many hypotheses, each dependent on the results of applying constraint rules, need to be run, tried, and discarded before a consistent set can converge. The computational and communication needs of these processes should be provided by high level processors which form the third tier of processing power needed to solve the image understanding problem.

2.3 Communication Between Processing Levels

Central to vision processing is the bi-directional flow of communication and control up and down through all representation and processing levels. These two capabilities allow us to perform multiple image operations, evaluate results of that processing, and re-compute with different parameters on different parts of the image. This must be done in tenths of milliseconds in order for different interpretation strategies to be tried dynamically within a single frame time.

In the upward direction, the communication consists of image abstraction and segmentation results from multiple algorithms, and possibly from multiple sensory sources. It also involves the communication of a set of attributes of each extracted image event to be stored in a symbolic representation. In addition, summary information and statistics allow processes at the higher levels to evaluate the success of lower level operations. In the downward direction the communication consists of knowledge directed processing and grouping operations, commands for selecting subsets of the image for specifying further processing in particular portions of the image, modification of parameters of lower level processes, and requests for additional information in terms of the intermediate representation.

Communications between levels may take four possible forms: One-to-one, one-to-many, many-to-one, and many-to-many. The first form involves one process at a given level communicating directly with another process at a higher or lower level. The second form is typically a process that broadcasts information to many lower level processes. The third form represents a collected feedback mechanism in which information from many lower level processes is combined and a result is passed to a single higher level process. The last form involves the parallel transmission of information between many processes at different levels. The associative processing paradigm, which will be presented next, provides for the first three of these communications mechanisms. The remaining mechanism will be discussed at the end of this section.

2.3.1 Associative Communications and Control

Based on our experience with highly parallel algorithms, [Levitan, 1986] we believe that the best way

to meet the requirements of high speed, fine grained communications and control is with associative processing techniques. There are three processing capabilities that are key to associative computation:

* Global Broadcast/Local Compare
* Some/None Response
* Count Responders

Associative processing can best be understood by an example of a single controller (a teacher) interacting with an associative array (a class of students) [Foster, 1976b]. If the teacher needs to know if any student in a class has a copy of a particular book the teacher can simply state, "If you have the book, raise your hand." The students each make a check, in parallel, and respond appropriately. This corresponds to a broadcast operation of a controller and a local comparison operation at each pixel in an array, to check for a particular value. Both operations assume that the local processors have some "intelligence" to perform the comparison.

Query and response is just the first part of associative processing. We have only described a content addressable ("If your hand is up, I'm talking to you.") scheme. To perform associative processing, we must be able to conditionally generate tags based on the value of data and use those tags for further processing. An example of this kind of association would be, "If the intensity of the red, green and blue values are each in a certain range, label yourself POSSIBLE-SKY"; certainly not a robust technique, but a colorful example. The interesting processing comes when the controller starts performing multiple logical operations on tags. Since each pixel (or region) could have multiple tags base on properties of the data as well as things like spatial coordinates, we could perform operations like: if you are NOT-TREE and NOT-ROOF and (POSSIBLE-SKY or POSSIBLE-CLOUD) and IN-TOP-OF-IMAGE then label yourself LIKELY-SKY. As processing continues only subsets of the pixels are involved in any particular operation. We are selectively processing pieces of the image based on their properties, but we are operating on all pixels with a given set of properties, in parallel.

The ability to associate tags with values is half the battle for high speed control. We also need to get responses back from the array quickly. Forcing the teacher to ask each student if they have their hand up defeats the process. The teacher can see immediately if any of the students have their hands up, and can quickly count how many do. Similarly, a Some-response/No-response (Some/None) wire running though the pixel array allows the controller to immediately determine properties about the data in the array, and therefore the state of processing in the array *without looking sequentially at the data values themselves.*

Additionally, fast hardware to perform a count of the responders allows the controller to see summary information about the state of the data in the array. We can write programs that can *conditionally* perform operations based on the state of the computation. By using the properties of the radix representation of numeric values in the array we can use the counting hardware to sum the values in the array. The ability to sum values gives us the power to compute statistical measures such as mean and variance.

These examples of students and pixels illustrate the power of associative processing. We use associative processing as our paradigm of communications in the upwards direction and control in the downwards direction between each pair of levels in the hierarchy. We broadcast criteria for selecting pixels, or regions, or symbolic tokens for selective processing. In this way higher levels control lower levels. We test and/or count the response that comes after processing data to allow conditional branching for the next step of processing in a given algorithm. Thus the lower levels provide feedback to higher ones.

The associative select operation also provides one mechanism for transferring non-summary information from a lower to a higher level in the vision processing hierarchy. The higher level may select a single value in a lower level and read it out. This transfer of information is one-to-one: a single value at a lower level is copied to a higher level.

2.3.2 Parallel Data Transfer Between Levels

Associative select is useful when a small number of data values must be copied to a higher level. When a large number of values must be transferred between levels a many-to-many data path between adjacent levels is more appropriate. A simple example is a data path that connects spatially collocated processors in adjacent pairs of levels. These processors may then perform inter-level transfers in parallel.

The many-to-many communications mechanism permits the use of a programming paradigm in which each level is used to transform a lower level representation into a higher level representation. All or part of this new representation may then be extracted by the next higher level of processing. Each level may then treat the level below it as an associative data base.

If the many-to-many communications mechanism also permits the passing of control information from higher to lower levels, then the granularity of control can be made to vary. This would allow, for example, initial processing to take place in a coarse-grained control mode, (SIMD) and later processing to take place in a fine-grained control mode (Multi-SIMD, or MIMD). The former is useful for generating initial hypotheses about an image, while the latter is better suited to resolving multiple local conflicts between those hypotheses, and to filling in details of the interpretations.

3. THE IMAGE UNDERSTANDING ARCHITECTURE(IUA)

The University of Massachusetts (UMass) together with Hughes Research Laboratories (HRL) is developing a three-level tightly-coupled associative architecture to encompass all levels of vision processing. The Image Understanding Architecture combines an integrated approach to the three types of computation outlined, including the critical problems of communication between the three levels.

3.1 Architecture Overview

The Image Understanding Architecture represents a hardware implementation of the three levels of abstraction in our view of computer vision. Overall it consists of three different, closely coupled parallel processors. These are the Content Addressable Array Parallel Processor (CAAPP)* at the low level, the Intermediate Communications Associative Processor (ICAP) at the intermediate level, and the Symbolic Processing Array (SPA) at the high level (figure 4).

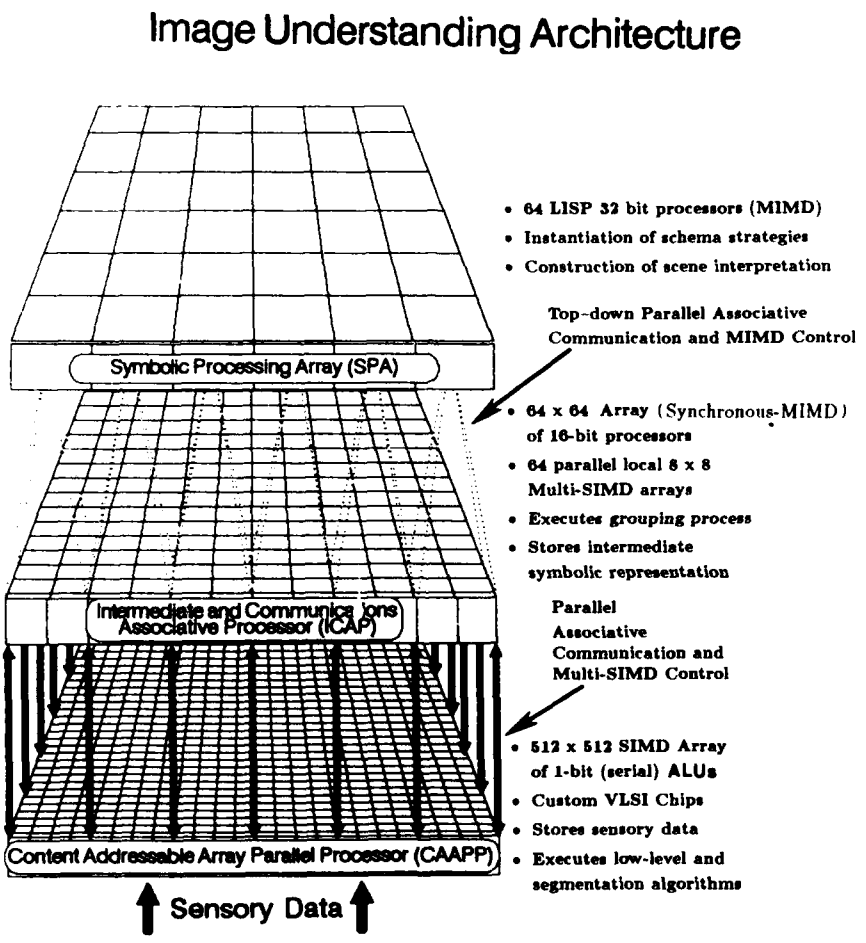


Figure 4. IUA Overview

* The term "content-addressable" is a synonym for "associative" and is an alternate term that now is not as widely used as it was when some of our work began [Weems, 1984a.]

The three levels of the IUA would at first appear to constitute a multi- resolution pyramid architecture,[Uhr, 1972; Tanimoto, 1983; Hanson and Riseman, 1980]. While it is possible to use the IUA to implement algorithms for a multi-resolution pyramid, this is not the use for which it is envisioned. Rather, the IUA implements a hierarchy of abstraction which corresponds to the three levels of abstraction (representation and processing) in the UMass VISIONS system.

Architecturally, it is easy to distinguish the differences between the IUA and a pyramid processor. In a multi-resolution pyramid, each higher layer is a power of two narrower than the layer below it, and all of the processing elements in the pyramid are identical, with each layer being treated as a SIMD parallel processor. In the IUA, however, the layer widths are 512, 64, and 8 - forming a sparse pyramid. Most importantly the processing elements in the IUA differ greatly from layer to layer. In each layer of the IUA the processing elements are tuned to the specific class of tasks required by that particular level of abstraction. For example, it is inappropriate to try to run LISP in parallel at the lowest level, because the low level is primarily concerned with fast pixel operations that will build an intermediate level symbolic representation. Thus, the low level processors are tuned for real-time image processing operations. At the highest level, on the other hand, symbolic AI processing will be the main objective, so the high level processors are tuned to run LISP code efficiently. This leads to a significant physical difference between the multi-resolution pyramid and the IUA: whereas the amount of circuitry in a pyramid decreases by a factor of four with each higher layer, the amount of circuitry in each layer of the IUA is roughly constant.

Another important difference between a multi-resolution pyramid and the IUA is that at the high level, the IUA is purely an MIMD parallel processor. Additionally, the intermediate and low levels of the IUA may be treated in a variety of modes. These include the CAAPP operating in pure SIMD or Multi-SIMD mode, and the ICAP operating in synchronous-MIMD or pure MIMD mode. In Multi-SIMD mode, the CAAPP cells excute in disjoint SIMD groups, with each group receiving a different instruction stream. In synchronous-MIMD mode, the programming paradigm is more like SIMD: The ICAP processors execute similar instruction streams, and globally synchronize for each stage of processing. Synchronous-MIMD has the advantage of being as simple to program as a SIMD system but without the time penalty of having to sequentialize on branching structures. The various modes of parallelism in the IUA are provided to allow multiple hypotheses from the SPA to be evaluated in parallel at the lower levels.

3.2 The Three Processing Levels

The CAAPP is a 512×512 square grid array of 1-bit serial processors intended to perform low-level image processing tasks. It is similar to the NASA/Goddard MPP [Batcher, 1980] but with an architecture that is especially oriented towards associative processing with global summary feedback mechanisms. This reflects the difference in application domain and processing strategy for these two machines. For example, a typical operation on the MPP involves enhancement of a large satellite image, where the results of the enhancement are presented to a human operator who then decides what further processing will be required. The goal of our work, on the other hand, is automated real-time vision without human intervention, where all of the processing and interpretation must be done by the system itself. Thus, the CAAPP has been tailored to permit flexible control, and to provide feedback to the controlling processes so that they may exercise control in response to actual image properties.

The intermediate level is implemented by the Intermediate and Communications Associative Processor (ICAP). The ICAP is also a square grid associative array, of more powerful processing elements; the ICAP is a 64 by 64 array of 16-bit processors. Each ICAP cell is associated with an 8 by 8 tile of CAAPP cells, to which it has access.

The ICAP is designed to manipulate the tokens in the Intermediate Symbolic Representation and to act as a data base for queries by processing elements in the SPA. For example, a house-roof schema in the high level may direct the ICAP to group together long, straight, parallel lines. The schema may then direct the ICAP to extract parallelograms that are candidate roof outlines.

A typical ICAP level symbolic representation of a line would consist of a unique label for the line and a set of fields which quantify its attributes. Such attributes may include end points, orientation, contour length, relative curvature, direction of curvature, average contrast across the line, labels of adjacent regions, nearby endpoints, and pointers to nearby or related lines. Clearly, while the bit-serial processors of the CAAPP are well suited for developing this representation, they will be inappropriate for manipulating it. Thus, the CAAPP is used to develop the intermediate symbolic representation, which is then passed to the ICAP for further processing. Should the need arise, the results of re-segmentation in the CAAPP can be integrated with the representation in the ICAP. To facilitate this processing capability, the ICAP representation is kept in approximate registration with the original image events in the CAAPP.

The SPA processors are powerful, general purpose microprocessors intended for performing high-level symbolic operations, and for controlling sub-array processing in the ICAP and CAAPP arrays. To the SPA, the lower levels appear as an intelligent database that is part of a shared global memory. The shared memory decouples the SPA processes from the locality of information in the image. An SPA process simply makes a request to the database and then waits for completion of the request before accessing the database to get the results.

The SPA processors will run a LISP-based blackboard system in which various schemas will cooperate and compete in the generation and verification of hypotheses about the content of the image and its relationship to models of the environment. From the point of view of the blackboard system,the CAAPP and ICAP will appear as knowledge sources at different levels of abstraction. The various schemas in the system will activate different processes in the CAAPP and ICAP for either the full array or independent sub-arrays.

The IUA is conceived as a stand-alone image interpretation system. Although a host processor is attached to the global controller for the IUA, the host is intended to serve the IUA rather than the other way around. The IUA host will provide a software development environment, and an access point for users of the IUA. The host may be used for examining the results of processing on the IUA, or for monitoring processes in the IUA. However, the host system does not take part in the actual image interpretation process. The dedicated global controller, which is an integral part of the IUA system, is responsible for managing the CAAPP and ICAP arrays in cooperation with the SPA.

3.3 Architecture Details

3.3.1 The CAAPP: Low Level Processing

The CAAPP consists of a 512 by 512 array of bit-serial processing elements that are linked through a four way (S,E,W,N) communications grid that is augmented with a proprietary circuit to allow certain types of long distance communication to take place quickly. Each processor will contain 320 bits of RAM, 5 one-bit registers (A, B, X, Y, Z), an ALU and data routing circuitry. Each element also has access to a 32K-bit backing store memory that is dual-ported with the ICAP. These processing elements are very simple, but it is this simplicity that allows us to place 64 of them on a single integrated circuit; a density of processing elements that is a prerequisite for constructing a machine with this many processors. It is this simplicity that also permits the CAAPP to execute instructions with a cycle time of 100 nanoseconds. The initial development of this design is decribed in great detail in [Weems 1984a]. Much work has gone into improvements to the CAAPP processing element since that time [Weems, 1984b, 1985].

The key to integrating the CAAPP into the IUA is its combination of associative feedback and control mechanisms. The principle feedback mechanism in the CAAPP is the array-wide logical OR output, called Some/None, which indicates whether any CAAPP cells are in a given state. At the end of each instruction cycle the logical OR of the response bit from every processing element is automatically available at two different levels. The global controller receives this signal for the full array, while the ICAP processors receive the Some/None indication for only the portion of the CAAPP array associated with them.

A count of all responding cells is also available at the global controller and ICAP level. The counting operation is used to gather statistics about an image and the results of processing. For example, through counting we may quickly determine the mean and standard deviation of an attribute for a given set of regions. The corresponding sub-array counts are available at the ICAP at the end of each CAAPP instruction cycle. The global controller develops the full count through a polling mechanism that takes 16 CAAPP instruction times to complete. However, the polling operation is independent of processing in the CAAPP cells and so the CAAPP may continue to operate while a count is being formed.

The Select-First operation is used to isolate a single CAAPP cell for readout or processing by the global controller, SPA, or ICAP processors. The primary purpose is to select single identified cells as representatives of groups (regions or segments) which can then be used to store facts about all cells in that group. For instance one cell in a region might keep the average color intensity and variance for the entire region. As necessary, the data in these cells can be moved up to the ICAP level.

The principle mechanism for transferring data between the CAAPP and ICAP is a shared memory structure. Besides the normal program and data memory, the ICAP has a 256K byte block of memory that is shared with the CAAPP. This 256K acts as a 32K bit by 64-bit backing store for the on-chip CAAPP memory and is the primary communications path between CAAPP and ICAP. Swapping to and from the backing store is done through dual-porting of a portion of the on-chip CAAPP memory. Although access to the off-chip memory is about 10 times slower for the CAAPP than access to the on-chip memory, the dual-porting arrangement permits double-buffered swapping to take place while the CAAPP is processing data in other memory segments.

In the event that large amounts of CAAPP data must be moved up to the SPA or controller, the data will first be transferred in parallel to the ICAP processors which are also linked to the SPA processors through a dual ported common memory.

The current design of the CAAPP processing elements has been achieved through four iterations of reduced instruction set analysis and redesign of the processing element architecture,[Weems, 1985]. The result is a design that is 60 percent faster than the original architecture, with an overall decrease in the device count for the processors and control circuitry. Figure 5 shows the CAAPP cell architecture, and figure 6 presents the instruction set for the CAAPP.

We are currently preparing a CMOS implementation of the CAAPP chip design with the cooperation of Hughes Research Laboratories. The 64 processing element chip

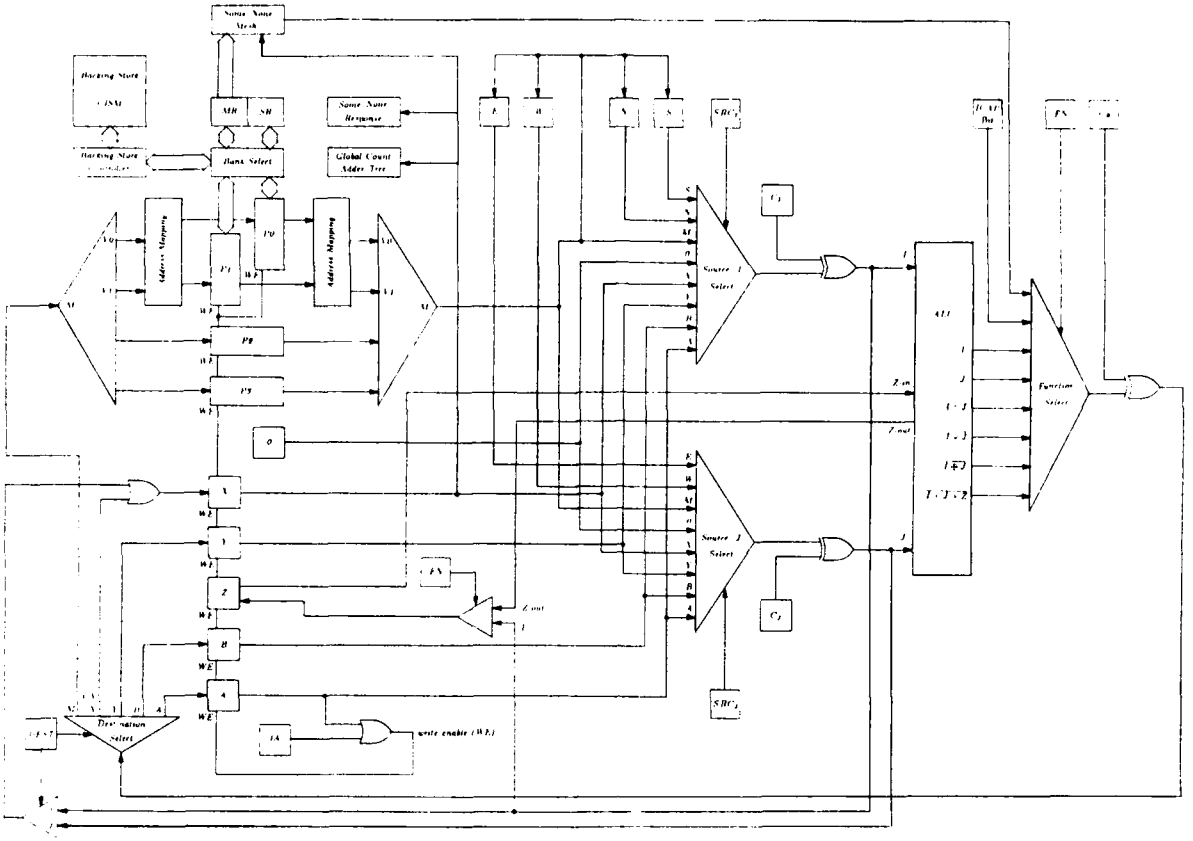


Figure 5. CAAPP Cell Architecture

is estimated to contain approximately 60,000 devices, of which 80 percent are in the on-chip memory. This is roughly the same number of devices found in 16-bit microprocessor chips, but is much simpler to design and implement because of the repetitive nature of the parallel processor cell design. It is essential that the primary working data memory be on the same chip as the processing elements, in order to maximize access speed, and to keep the number of pins on the chip within reasonable limits.

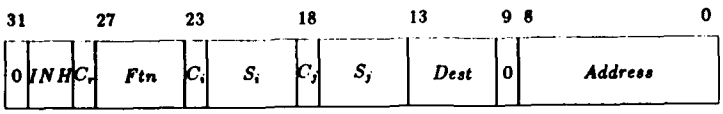
The CAAPP algorithm in figure 7 demonstrates how the CAAPP is programmed, and the importance of rapid feedback from the CAAPP to its controller. This algorithm selects all cells that contain the maximum value in a given field. In addition, the maximum value is available in the controller at the end of this operation. Selecting maximum values requires three CAAPP instruction cycles for each bit in the field.

The algorithm begins by loading the high order bit of a field into the response register of all active cells. The global controller then tests the Some/None output of the array. If any cells have their high order bit set, then they are candidates for the maximum value. Any cells that have a zero in their high order bit are then deactivated. However, if no cells have their high order bit set, then none are deactivated because they are all still potential candidates. This process repeats with each successively lower order bit in the field. When the low order bit has been processed, only those cells which contain the maximum value will remain active. For each iteration, the controller saves the Some/None response so that the maximum value is available in the controller at the conclusion of processing.

3.3.2 The ICAP: Intermediate Level Processing

The ICAP is also a square grid array processor, and is intended to perform intermediate level processing on image events that are in physical registration with the pixel data. Each ICAP processing element is associated with an 8 by 8 tile of processors in the CAAPP array. An ICAP processor has access to data stored in any of the 64 CAAPP processors it is associated with. Each ICAP processor also has access to the global summary information for those 64 processors.

There are two reasons for choosing 64 as the number of CAAPP processors to associate with each ICAP processor. First, it is convenient to associate one ICAP with each



INH	(Inhibit)
0	Non-inhibit — always active
1	Inhibit if A = 0
2	Inhibit if A = 0 or S/N = Some
3	Inhibit if A = 0 or S/N = None

$C_{i,j,r}$	
0	TRUE
1	Complement $S_{i,j}$, *Result*

	S_i	S_j	*Ftn*	*Dest*	
0	zero-reg	zero-reg	$Some/None \Rightarrow Dest$	X_{pr}	0
1	C	C	$S_i \Rightarrow Dest$	A, X	1
2	S	E	$S_j \Rightarrow Dest$	$A, X \Leftarrow S_i$	2
3	N	W	$\overline{S_i \bigwedge S_j} \Rightarrow Dest$	$A, X \Leftarrow S_j$	3
4	Y	Y	$\overline{S_i \bigvee S_j} \Rightarrow Dest$	Y	4
5	X	X	$\overline{S_i \bigoplus S_j} \Rightarrow Dest$	X	5
6	B	B	$\overline{S_i + S_j + Z} \Rightarrow Dest$	B	6
7	A	A	$ICAP\ C \Rightarrow Dest$	A	7
8	memory	memory	$S_i \Rightarrow Z$	memory	8
9	—	—	$memory \Rightarrow MR$	—	9
10	—	—	$memory \Rightarrow MR, SB$	—	10
11	—	—	$MR \Rightarrow memory$	—	11
12	—	—	$MR, SB \Rightarrow memory$	—	12
13	—	—	—	—	13
14	—	—	—	—	14
15	—	—	—	—	15

Figure 6. CAAPP Cell Instructions

```
FOR Bit Num :  Field Length - 1 DOWNTO 0 DO
    Response := Field[Bit Num]
    IF Some
      THEN
        Activity := Response
```

Figure 7. Finding a Maximum

CAAPP integrated circuit, since this greatly simplifies the interface between the two levels. Second, with a 512 by 512 array at the bottom and three processing levels plus a single controller at the top, a uniform inter-level connectivity is provided by a 64 to 1 reduction factor between each of the levels. Finally, given the processing needs and increased power of the processing elements at each level, we roughly estimate 64 as a viable scale reduction between each pair of levels.

Control of the ICAP is provided by the global controller (in Synchronous-MIMD mode) and by the SPA (in MIMD mode). Once an intermediate symbolic representation has been passed to the ICAP, and initial grouping operations have taken place, each of the SPA processors may then query the ICAP in parallel to establish and verify hypotheses. The ICAP provides four different global OR responses and a global summation value as feedback to the global controller.

Each of the 4096 (64 by 64) ICAP processors consists of an ALU that may perform 16, 8, or 1 bit operations, 256k bytes of local RAM, dual ported memory for interacting with the CAAPP and SPA, and neighbor communications hardware. The hardware for an ICAP cell consists of an off-the-shelf CPU and memory, plus a custom VLSI circuit that provides inter-level communications and control functions. For our ICAP processor we have chosen the TMS320C25. The 320C25 operates at 5 million instructions per second and can perform a 16 bit multiply and accumulate operation in a single instruction time.

The ICAP communicates in the vertical dimension via two sets of shared memory structures. There is a shared memory which acts as a backing store for the CAAPP and is available in the address space of the ICAP. In addition, the ICAP and the SPA share a common memory. This is viewed as an I/O device for the ICAP and is visible in the

address space of all the processors of the SPA. Communication between the ICAP and the SPA takes place in much the same way as communication between the CAAPP and ICAP except that they are transferring information that is represented more abstractly than the information that is passed between the CAAPP and ICAP. These up and down connections provide the bi-directional and inter-level communications necessary for image interpretation. Through the up/down links, information may be passed upward as its level of abstraction dictates, and reinforcements of competing hypotheses may be passed downward as needed to resolve conflicts.

The side to side links in the ICAP provide the intra-level communications necessary for grouping and merging processes to take place on the intermediate symbolic representation. These links are circuit switched and provide a 5 M-bit/second data transfer rate between ICAP processors. The topology of this network is a mesh with additional links that provide for both local neighborhood and long distance communication across the array.

3.3.3 The SPA: High Level Processing

The high level is implemented by the Symbolic Processing Array (SPA), which is an ensemble of 64 powerful processors. Each SPA processor is a 32-bit computer with at least 4 Mbytes of memory. In the first prototype slice of the IUA, the SPA will be an M68020 class processor.

The SPA is designed to support parallel rule-based and schema (frame-based) processing within the framework of a blackboard architecture [Draper et al., 1986]. In such a system, various schemas are activated in parallel via focus-of-attention strategies to evaluate results of initial bottom-up processing. These schemas generate hypotheses which activate verification processes in the form of other schemas. These, in turn, take MSIMD control of the ICAP and CAAPP to perform top-down processing and re-evaluation. The schema processes cooperate and compete via messages posted on the system blackboard.

Under our current plan, communications between SPA processors will take place in part via a virtual full-connection network. This will be implemented by a high-speed ring structure that allows one data word to be passed from each processor to every other processor in a single instruction time. The ring I/O ports into each processor will be memory mapped into a dual ported content addressable input/output memory (CAIOM). This structure was previously described [Levitan, 1984], as a Full-CAPP interconnect. Such a structure is particularly well suited for implementing a distributed blackboard system. Unlike a simple full-connect network, the CAIOM augmented full connect network does not overload the processors with having to process all of the messages that have been transmitted. Because the messages are stored directly into an associative memory upon arrival, the processor may examine them via parallel associative operations. In many cases, the Some/None and Count values are all the information about the messages that is needed. If a particular message must be examined directly then the Select-First mechanism permits the SPA processor to directly access that message, without first having to search through the other messages. The CAIOM is viewed as a high priority mechanism for passing update and control information about the state of the blackboard. The blackboard itself will most likely reside in a global shared memory that includes a data space which is dual-ported with the ICAP processors. This approach allows the blackboard to contain a tremendous amount of information and yet the knowledge source's attention can be quickly refocussed through message passing over the CAIOM network.

3.4 Architecture Summary

The three-level structure of the UMass Image Understanding Architecture supports the necessary hierarchy of abstractions for the different representations and operations we need to solve the vision problem. Each level is constructed to perform a suite of tasks most appropriate for that level of abstraction.

The CAAPP is optimized to perform local operations on neighborhoods of pixels and to provide feedback to the higher levels of processing about the state of the computation and statistics about low level data. It excels at very tightly-coupled fine-grained parallelism. The mapping of one pixel onto each processor ensures that the maximum amount of parallelism available in the low level vision tasks will be utilized.

The ICAP is designed to support the necessary tasks of building an intermediate representation of the image and operating on that representation. These operations need two primary capabilities, data manipulation and communication. The data representations used by the CAAPP need to be transformed by the ICAP into a more accessible format, and then passed to neighboring ICAP cells to perform merging and grouping operations.

The high level tasks which perform schema-based reasoning run in the SPA. Schemas are processes consisting of programs and data structures which are used to reason about the image in terms of a knowledge database. To support this kind of distributed artificial intelligence processing we need powerful processors with large amounts of memory. The communication between processes will primarily be in terms of a distributed blackboard system managed by the processes themselves. As these processes run and make requests to the ICAP (and sometimes directly to the CAAPP) they will extract information about the image and post the results of their analysis on the blackboard for other processes to use. The end result will be an interpretation of the image.

3.5 Current Status

At present we are building a 1/64th scale demonstration prototype of the IUA. This is scheduled for completion in early 1988 and will include 4096 CAAPP cells, 64 ICAP cells, a single SPA processor and an array control unit. The entire prototype will plug into a single-user workstation that will serve as a host.

The prototype will physically consist of a 16 by 20 inch mother board with 83 daughter boards. Eighty of the daughter boards are 4 by 2.5 inches and the remaining three are 20 by 2.5 inches in size. Of the 80 small daughter boards, 16 are for clock distribution and signal buffering. The other 64 contain the ICAP and CAAPP processors and their memories. The three larger daughterboards provide the controller interface, feedback concentration, and ICAP communications network switching. The motherboard also includes a dual ported frame buffer memory that allows high speed image input and output.

Each processor daughterboard will contain a single custom VLSI chip, TMS320C25, 256K bytes of static RAM, 384K bytes of dual-ported dynamic ram, and tri-state bus buffers. The single custom chip holds the 64 CAAPP processors with their local memories, the backing store controller, a refresh controller for the dynamic RAM, and arbitration logic for the various devices that must access the bus of the associated ICAP processor. Total power dissipation for a processor daughterboard is estimated at approximately 5 watts.

The custom VLSI chip is being designed in 2 micron CMOS with 2 metal layers. A 32 processor test chip is currently undergoing fabrication through the MOSIS facility. A first run of the complete custom chip is scheduled for summer of 1987.

Our software simulator is being re-written to run on an Odyssey signal coprocessor in a Texas Instruments Explorer. The Odyssey allows a direct emulation of the ICAP processor and greatly improves the execution times for CAAPP simulations over our VAX based simulator. The Odyssey simulator will also permit us to closely mimic the interactions of the three processing levels down to the signal level. The Odyssey based simulator will initially provide the capability of a single IUA daughterboard, and will eventually be extended to simulate one motherboard.

A VAX-based high level emulator is also planned for development. Whereas the Odyssey simulator is designed to allow assembly language level programming, the VAX emulator will be the vehicle of choice for researchers who wish to get an idea of how the user-level IUA environment will behave. The emulator will sacrifice low level accuracy in favor of greater speed. For example, the emulator will be restricted to 8, 16 and 32 bit arithmetic, avoiding the slow-to-simulate bit-serial methods that are actually used in the CAAPP.

Beyond simply testing our hardware design, our ultimate goal for the prototype is to provide a powerful interim development environment for image understanding parallel processing research. A simulated parallel processor is simply too slow to permit any significant amount of experimentation. Once our prototype is up and running, we will be able to accomplish more in the first ten seconds of execution time than we have been able to do in our previous five years of simulation.

Because having this much processing power in a box the size of a personal computer is so attractive, we have designed our prototype to be easily reproducible for a reasonable cost. It has also been designed to be easily adapted to different host systems. We thus hope that it will be possible to construct several copies of the small scale system so that it can be available to a number of researchers prior to construction of the full scale machine.

4. THE OPERATING ENVIRONMENT

An integrated multiprocessor system cannot be programmed as a collection of separate machines. For our system, even the three tiers of processors must be viewed as a single system. This does not make programming harder, rather this should be viewed as a version of the classic model of a layered system. In operating system design, for instance, we often design a system as a hierarchy of virtual machines – each providing a service to the one above by making demands (calls) to the one below. The IUA follows this paradigm explicitly: The hardware / software system at each level provides a clean virtual machine for the levels above and below it.

The high level system needs to run a parallel LISP implementation of the schema system. The currently existing software environment for schema development on a single processor is outlined in [Draper et al., 1986]. Besides the necessary support for a parallel LISP environment, we need special-purpose hardware (and system support) for two functions: inter-schema communication and queries and updates to the intermediate level representation. Schemas not only need to communicate with other schemas to resolve conflicts and ambiguity, they need to locate which other schemas, out of hundreds or thousands, share related image events. The CAIOM hardware, and LISP/operating system support will allow schemas to post requests and associatively search for other schemas with the same objects of interest in the image.

Queries to the intermediate level from the SPA are processed by making requests to the ICAP processors. These requests can be of two forms, local to a specific sub-image, or global to the whole image. The global array control unit receives these requests and queues them for processing. Processing a request involves running one (or several) "scripts". These are microcode sequences broadcast to the CAAPP/ICAP arrays as sets of instructions and data. The

scripts are not free of loops or branches. Close interaction with the feedback from the arrays makes it possible to program data-dependent conditional execution in the scripts. Data dependent interactions between the ICAP processors and their associated CAAPP cells is also possible.

5. THE DEVELOPMENT ENVIRONMENT

During the hardware development phase, software functional level simulators are being used for testing implementations of vision algorithms on the IUA. CAAPP simulators have been in use to refine both the architecture and low level algorithms. New CAAPP and ICAP simulators will be available both separately, for testing algorithms that run on a single level, and in an integrated form. The SPA simulator will be a multiprocessing CommonLisp environment, running on a commercial multiprocessor, and augmented with functions that simulate the CAIOM communications mechanism.

The simulators may be adjusted to simulate a CAAPP with a width of any power of two from 8 through 512 (with the ICAP simulator adjusted correspondingly to a width of 1 through 64). Most software testing will be done with the simulators set to a small array width to facilitate turnaround time on simulations. When the simulators are set to full width, our experience has been that we achieve less than one second of simulated IUA time per year of VAX CPU time. Of course, once the hardware becomes available, the development environment will shift entirely onto it, in order to speed up software testing.

In addition to the simulators, assemblers are planned for the CAAPP and ICAP. These are assemblers only in that they allow the programmer to program the CAAPP and ICAP at the instruction-by-instruction level. The assemblers will be interactive, providing information to the programmer that will aid in optimizing utilization of the two levels in parallel. Programs that are constructed through the assemblers will initially execute on the simulators, and later on the actual hardware. Each program generated by the assemblers also takes the form of a LISP or C callable subroutine. This permits the SPA simulator, and the UMass VISIONS system to interface directly with any software that is developed for the IUA.

6. FURTHER DEVELOPMENT

We are currently working with researchers at Hughes Research Laboratories on refinements to the ICAP structure and the CAAPP to ICAP control interface. One possibility being explored is to give the ICAP the ability of interacting with a local CAAPP controller. The CAAPP would then be able to execute in Multi-SIMD mode at the chip level: Each 8×8 section could then execute a separate instruction stream. Another extension would be to augment the ICAP globally controlled circuit switched network with a distributed control routing network.

7. CONCLUSION

As we discussed in section 1, we believe that to build a computer vision system we must provide an architecture which can help address the problem areas of: the unreliability of segmentation processes, the compounding of uncertainty from different processing techniques, the need to provide mechanisms for information fusion as well as support for multiple representations of data and meta-knowledge, the need for global context dependent local processing, and support for inferencing techniques.

The Image Understanding Architecture consists of an integrated vision system executing on a parallel processing ensemble. It supports many processes performing many tasks at three levels of abstraction. Each level is embodied in a particular set of processing elements suited to the requirements at that level of processsing. High speed low level local processing elements provide fast segmentation processes which can be tuned and retried. Associative processing allows feedback from the processes to the control system as well as supporting context dependent operations. Information fusion is provided by intermediate level processes running at the appropriate level of abstraction to reduce uncertainty from different processing strategies. Symbolic processing at the high level supports a schema based inference system where knowledge about the real world is represented as both data and control information. Communication and control both between levels (vertically) and among processors at each level (horizontally) allows the system to work in a concerted manner to perform machine vision.

8. ACKNOWLEDGEMENTS

We would like to thank our colleagues at Hughes Research Laboratories in Malibu, Dr. J. Gregory Nash and Dr. David B. Shu for their many contributions to the design and implementation of the Image Understanding Architecture.

This research was supported, in part, by the Advanced Research Projects Agency of the Department of Defense and was monitored by the Air Force Office of Scientific Research under Contract No. F49620-86-C-0041 and by the Engineer Topographic Laboratory under Contract No. DACA76-86-C0015. We would also like to thank Caxton C. Foster for his ideas, support and guidance.

9. REFERENCES

Akers, S. B. and Krishnamurthy, B., A Group Theoretical Model for Symmetric Interconnection Networks, *Proc. 1986 International Conference on Parallel Processing*, K. Hwang, S. M. Jacobs and E. E. Swartzlander (Eds.), St. Charles, Ill., August 19-22, 1986, pp. 216-223.

Batcher, K. E., Design of a Massively Parallel Processor, *IEEE Trans. Comp.*,29:1-9, September 1980.

Draper, B., Hanson, A., and Riseman, E., A Software Environment for High Level Vision, COINS Technical Report, University of Massachusetts at Amherst, (in preparation) 1986.

Erman, L., et al., The Hearsay-II Speech-Understanding System: Integrating Knowledge to Resolve Uncertainty, *Computing Surveys*, 12:213-253, 1980.

Faugeras, O., and Price, K., Semantic Descriptions of Aerial Images Using Stochastic Labeling, *IEEE Trans. Pattern Analysis and Machine Intelligence*, 3:638-642, 1981.

Foster, C. C., *Content Addressable Parallel Processors*, New York: Van Nostrand Reinhold, 1976. (a)

Foster, C. C., *Computer Architecture*, New York: Van Nostrand Reinhold, 1976. (b)

Hanson, A. R. and Riseman, E. M., Preprocessing Cones: A Computation Structure for Scene Analysis, COINS Technical Report 74C-7, University of Massachusetts at Amherst, September 1974.

Hanson, A. R. and Riseman, E. M. (Eds.), *Computer Vision Systems*, New York: Academic Press, 1978.(a)

Hanson, A. R., Riseman, E. M., VISIONS: A Computer System for Interpreting Scenes. In: *Computer Vision Systems*, A. R. Hanson, and E. M. Riseman (Eds.), New York: Academic Press, 1978, pp. 303-333.(b)

Hanson, A. R., Riseman, E. M., Segmentation of Natural Scenes. In: *Computer Vision Systems*, A. R. Hanson, and E. M. Riseman (Eds.), New York: Academic Press, 1978, pp. 129-163.(c)

Hanson, A. R., Riseman, E. M., Processing Cones:A Computational Structure for Scene Analysis for Image Analysis. In: *Structured Computer Vision*, S. Tanimoto and A. Klinger (Eds.), New York: Academic Press, 1980.

Hanson, A. R., Riseman, E. M., A Methodology for the Development of General Knowledge-Based Vision Systems, (to appear). In: *Vision, Brain, and Cooperative Computation*, M. Arbib and A. Hanson (Eds.), Cambridge, MA: MIT Press, 1986.

Levitan, S. P., *Parallel Algorithms and Architectures: A Programmers Perspective*, Ph.D. Dissertation, Computer and Information Science Department, also, COINS Technical Report 84-11, University of Massachusetts at Amherst, May 1984.

Levitan, S. P., Measuring Communication Structures in Parallel Architectures and Algorithms, (to appear). In: *The Characteristics of Parallel Algorithms*, L. Jamieson, D. Gannon, and R. Douglass (Eds.), Cambridge, MA: MIT Press, 1987.

Parma, C. C., Hanson A. R., and Riseman, E. M., Experiments in Schema-Driven Interpretation of a Natural Scene, COINS Technical Report 80-10, University of Massachusetts at Amherst, April 1980. Also in: *NATO Advanced Study Institute on Digital Image Processing*, R. Haralick and J. C. Simon (Eds.), Bonas, France, 1980.

Reynolds, G., Irwin, N., Hanson A. R., and Riseman, E. M., Hierarchical Knowledge-Directed Object Extraction Using a Combined Region and Line Representation, *Proc. of the Workshop on Computer Vision: Representation and Control*, Annapolis, Maryland, April 30 - May 2, 1984, pp. 238-247.

Tanimoto, S., A Pyramidal Approach to Parallel Processing, *Proc. 10th Annual International Symp. on Computer Architecture*, Stockholm, Sweden, June, 1983.

Uhr, L., Layered Recognition Cone Networks that Preprocess, Classify and Describe, *IEEE Trans. Comp.*,21:758-768, 1972.

Weems, C. C., *Image Processing on a Content Addressable Array Parallel Processor*, Ph.D. Dissertation Computer and Information Science Department, also, COINS Technical Report 84-14, University of Massachusetts at Amherst, September 1984.(a)

Weems, C. C., Levitan, S. P., Foster, C. C., Riseman, E. M., Lawton, D. T., and Hanson, A. R., Development and Construction of a Content Addressable Array Parallel Processor (CAAPP) for Knowledge-Based Image Interpretation, *Proc. Workshop on Algorithm-Guided Parallel Architectures for Automatic Target Recognition*, Leesburg, VA, July 16-18, 1984, pp. 329-359.(b)

Weems, C. C., The Content Addressable Array Parallel Processor: Architectural Evaluation and Enhancement, *Proc IEEE International Conference on Computer Design: VLSI in Computers*, Port Chester, New York, October 7-10, 1985. pp. 500-503.

Weymouth, T. E., *Using Object Descriptions in a Schema Network for Machine Vision*, Ph.D. Dissertation, Computer and Information Science Department, also, COINS Technical Report 86-24, University of Massachusetts at Amherst, 1986.

CONSTRUCTS FOR COOPERATIVE IMAGE UNDERSTANDING ENVIRONMENTS

Christopher C. McConnell, Philip C. Nelson, Daryl T. Lawton

Advanced Decision Systems, Mountain View, California 94040

ABSTRACT

Researchers in machine vision require a rich set of tools for developing theories, algorithms, and systems. These range from supporting the expression of numerically intensive computations to rule-based, symbolic inference processes. This paper describes a tightly coupled set of such constructs that have been developed for a software environment for image understanding. The major components of this are: 1) an object-oriented programming mechanism for defining, creating, modifying, and combining objects. There are also databases for the storage and access of long term information such as procedures, object definitions, and instances of processing environments; 2) An underlying functional form built from an extendable set of macros that is used for expressing common processing operations for several different types of objects; 3) A set of uniform databases for automatically maintaining resources and results in an interactive or autonomous processing environment; 4) An interactive display facility for images that generalizes to a wide range of other objects, including curves, regions, and groups.

1. INTRODUCTION

Researchers in computer vision deal with problems at several different levels of computation, from numerically intensive computations that are often accomplished in parallel over large, structured arrays, to processing involving symbolic and relational objects, such as extracted image structures, instantiated model hypotheses, and interpretation rules. This work requires a highly interactive environment in which to prototype, experiment, and interpret results. In addition, work in computer vision usually involves researach teams, because of the range of problems being addressed. It is essential, especially in building application systems of any complexity, that tools support the integration and capabilities of several people's work. The set of tools and the programming environment dramatically shapes the culture for doing vision research Donaldson et.al. - 83, [Kohler et.al. - 83], [Quam - 84].

Figure 1-1 shows the basic components that have been developed as part of such an environment. The object/function definition facility supplies a powerful and uniform set of constructs for creating object types and instances at several of the different levels necessary for vision research. These include image registered features, intermediate objects such as regions, and high level objects such as visual models. The display facility provides a uniform mechanism for viewing objects.

There are four databases for automatically keeping track of created object instances (Perceptual Structure Database or PSDB), processing history (Environmental Database or EDB), existing functions and object type definitions Programmers Database or PDB), and stored results such as images, objects, and particular instances of the PSDB, EDB, and LTDB. These databases are accessed through a uniform query language or, optionally through a set of interactive browsers. The database mechanisms allow research techniques and results to be shared by a distributed community of users.

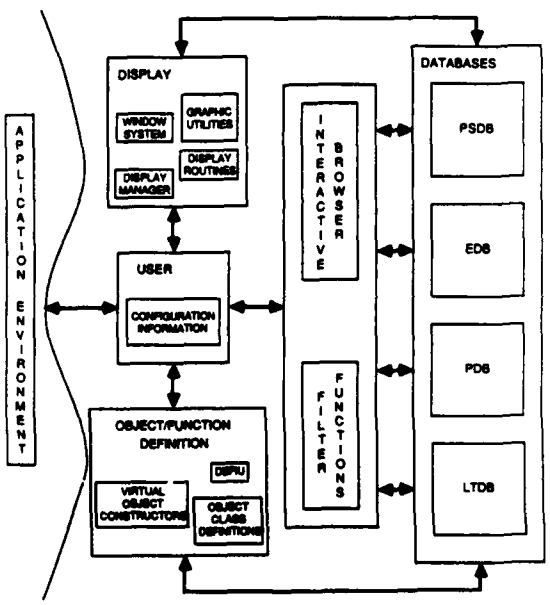


Figure 1-1: Components

2. OBJECTS

It is necessary to represent many different types of objects in computer vision. To function within an integrated environment, the method of defining, saving, accessing, displaying, and combining these objects must be specified. In addition to these common features, an object must also represent its class specific properties (e.g., all points have a location), and perhaps some instance specific properties (e.g., this curve has this interesting feature). The ability to assign both class and instance attributes to objects allows a researcher to standardize well understood properties, but still permits a rich, dynamic representation mechanism.

To meet these and other requirements, objects are implemented in an object-oriented programming methodology. In such a system, an object and the procedural forms used to manipulate it are symbolically attached. An object is then manipulated by instructing it to evaluate the symbolically specified operation - known as sending the object a message. This is best illustrated by an example as follows: In standard programming, a display routine that requires the location of an object (i.e., a point) would access the value with a function available in the environment, (location <point>). In an object-oriented system, the points location would be queried as (send <point> :location).

When only one type of object is present in the environment, these two approaches are merely a syntactic transformation. A problem arises, however, with the introduction of new types of objects. The location of a curve, for example, may be defined as the geometric center of its bounding box as opposed to the row and column associated with a particular point. To enable the display routine to manipulate curve objects as well as points, the location function must be modified to act differently depending on the type of its input. The need to constantly modify the manipulators to support new objects, quickly yields a bulky, inflexible, unorganized system.

In the object-oriented example, the curve object symbolically attaches the bounding box calculation to the location message, completely independent of the characteristics of a point. The task of specifying how an object is to be manipulated is distributed out of the overloaded location operator and is transferred into the object. The object now determines the implementation of the semantic property, "location".

This distribution of functionality gives a great deal of power. From the caller's perspective, all objects are seen to support similar properties, while objects need only be concerned with their own manipulators. Multiple representations of the same object type can be supported as well as multiple types. Because instances in our environment can contain their own specific properties, a particular object can interpret a message differently than all others in its class. This capability is the foundation of virtual objects, described later in this section.

2.1 DEFINED PERCEPTUAL OBJECTS

Several types of objects in the initial environment have been defined, including images, points, curves, and regions. These objects support the basic properties common to all objects: save-to-file, copy, and add - tional properties particular to each class. There are also composite objects, stacks, and groups that abstractly combine collections of other objects.

Interesting combinations of the initial objects are immediately apparent. An image can be created where each pixel is a pointer to a list of extracted objects that exist at that location. This is called a label plane and is used extensively for geometric reasoning. A region can be bounded by a list of curves, and a group can enclose similar regions. Any desired combination is possible.

An important feature of all non-image objects such as points, junctions, curves, and regions, is the representation of their spatial characteristics. Any representation should be compact, provide fast access, and should facilitate most common operations. However, there is no optimal format; each must trade off between time and space considerations.

The initially defined objects allow the researcher to choose among several possible pre-defined representations: arrays, segments, chain codes, and quad trees. These variations are possible with our utilization of object-oriented programming. When a representation is chosen, the object inherits the procedures associated with the relevant messages, and the newly defined object can immediately be manipulated in the environment.

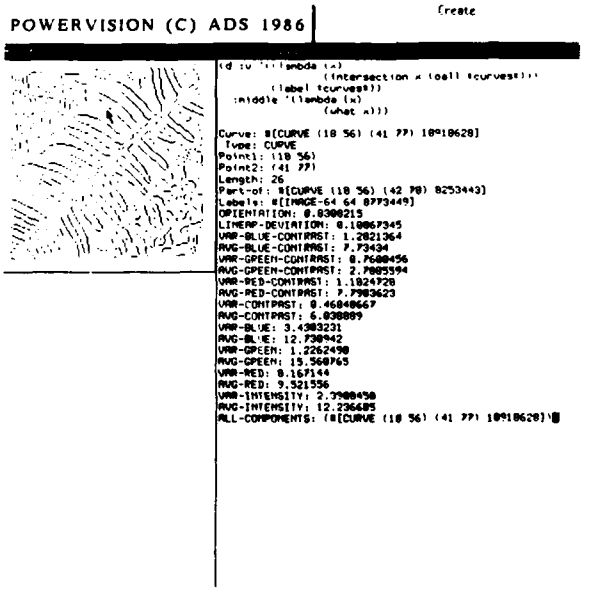


Figure 2-1: Interactions With Label Plane

As an example of label planes and system defined object types, Figure 2-1 shows a display function called display-values for interactively accessing and applying functions to values in images. In this figure, the image is a label plane for the set of extracted curves and the result returned by the function is the description of the curve instance that is selected. The default curve attributes are shown in small letters and the associatec attributes are shown in capitals.

2.2 OBJECT ACCESS

As discussed above, all objects in the environment are accessible through a uniform set of messages through the object-oriented programming methodology. This layer of procedural abstraction allows powerful generalizations. A small set of display routines can be applied to a large number of varied objects. The messages can mask any complexity in describing an object's physical characteristics. Virtual objects can be created that have no internal data but are procedural filters over other objects already defined.

The uniform interface requires each object to understand messages in both a 1D and 2D context. This allows the system to make no distinction between objects that are considered inherently "linear" or inherently "planar". There are three basic messages in the uniform object interface each with a one and two dimensional version:

1. BOUNDARIES -
   - 1D) The number of data points in the object.
   - 2D) The minimum and maximum X and Y value in the object.
2. POSITION -
   - 1D) Maps an X,Y position into the equivalent 1D offset.
   - 2D) Maps a 1D offset into the appropriate X and Y value.
3. VALUE -
   - 1D) Returns the color at a particular offset.
   - 2D) Returns the color at a particular X and Y position.

To put these messages into perspective, consider a two dimensional image object. The meanings of the 2D messages is clear. The BOUNDARIES message returns the dimensions of the array, the POSITION message returns the offset of a specified row and column from the base of the array, and the VALUE message references the pixel value at a particular row and column. The 1D messages can be interpreted as follows: the BOUNDARY message returns the number of pixels in the image (i.e., Width * Height), the POSITION message does the inverse mapping from the array offset to the appropriate row and column, and the VALUE message does a lookup treating the array as a very long 1D vector, similar to the way it is stored in memory.

The meaning of the 1D and 2D messages are not as correlated in a curve object represented as a list of (X, Y, COLOR) triplets. The 1D BOUNDARIES messages returns the total number of points in the curve, the 1D POSITION message returns the X and Y position of the nth point, and the 1D VALUE message returns the color associated with the nth point. The 2D messages can be interpreted as follows: the BOUNDARIES message returns the geometric bounding box (i.e., minimum and maximum X and Y values) of the curve, the POSITION message returns the point number of the point at, or closest to, a particular X and Y, and the VALUE message calls the 2D POSITION message to find the appropriate point and then calls the 1D VALUE message to look up its color. This is an example of combining messages to perform a complex action.

Note that this is not the only interpretation for a curve object. For example, the 2D POSITION message may only interpolate to the closest point when the specified X and Y lie along the curve boundary. The 2D VALUE message may interploate the colors of the two adjacent points rather than returning the color of the closest point. This is a major advantage of the functional interface. An object decides how it's to be described; its description is NOT determined by any environmental routines.

This system also supports multiple internal representations. If a curve is associated with a label plane, the curve object can reference the label plane to answer the 2D messages and the (X,Y,COLOR) list to answer the 1D messages. Again, this capability is independent of the interface system, it is wholly contained inside the object.

It should be noted that an object need not understand every message. The drawbacks of such an incomplete object are that it may not function properly in some display routines, and it may not combine with other objects or other procedural filters to create compound or virtual objects.

2.3 VIRTUAL OBJECTS

A virtual object looks and acts like any other object in the environment; however, the procedures underlying its access messages modify the descriptions of other objects without duplicating their underlying representation. In other words, a virtual object is just a filter around the messages of other objects. A simple example is a virtual image (V-IMAGE) which is the inverse of an existing image (IMAGE). V-IMAGE uses the BOUNDARIES and POSITION message from IMAGE unchanged, but modifiies the 1D-VALUE message as follows:

```
'(lambda (index)
  (logxor -1 (send IMAGE :1D-VALUE index)))
```

There is a speed disadvantage in this virtual object because the XOR calculation must be done each time a pixel is accessed. However,there is tremendous space efficiency because V-IMAGE, a unique, new image, is created at the cost of a few lambda expressions.

Virtual objects are especially important in a display system where many minor modifications of existing objects are desired to tune the appearance of the display. Rather than having a fixed set of these tuning options in the display routines, such as scaling, inverting, adding, etc., any combination that can be expressed as a lambda expression can be specified. Binary masks can be used to conditionally modify the value of a full 8-bit image. A label plane of pointers to objects, that is not a directly displayable image, can be run through a filter that returns an integer based on the attributes of the object pointed at in the label plane in any given position. Because a virtual object is itself a complete object, answering all the messages specified in the uniform interface, it can be used recursively as a data source in a new virtual object.

Virtual objects also provide an excellent interface to various types of display hardware. An 8-bit image can be displayed on a monochrome display by mapping some values to one, and the rest to zero. Three 8-bit images can be combined into a 24-bit virtual image by adding together the first image value, the second image value shifted by 8 bits and the third image value shifted by 16 bits.

Virtual objects can also map over localized neighborhoods in the source object. Gradients, convolution, and median filtering are all possible - without ever creating a new image. An example is a virtual image that returns the maximum value in a 2 by 2 neighborhood in a source image. The 2D-VALUE message of such a virtual image is:

```
'(lambda (row col)
  (max (send IMAGE :2D-VALUE   row      col)
       (send IMAGE :2D-VALUE (1- row)   col)
       (send IMAGE :2D-VALUE   row   (1- col))
       (send IMAGE :2D-VALUE (1- row) (1- col))
       ))
```

Referencing a pixel in this virtual image is at least four times as expensive as referencing a pixel in the source image but again, there is a tremendous space savings. Note that the virtual image described above is actually one pixel smaller than the source image (i.e., the source image must always have at least one row and one column beyond any call to 2D-VALUE). This is accomplished by filtering the BOUNDARIES message of the original image through a function that reduces the dimensions by one.

As demonstrated in the above example, virtual objects are not restricted to modifying only the VALUES message of their source objects. By modifying the POSITION and BOUNDARIES messages, other powerful effects are possible. For example, to reduce an image by a factor of two by averaging local neighborhoods, the BOUNDARIES message divides the orginal bounds by 2, the POSITION message maps from reduced pixel locations to their original position (by multiplying the row and column values by 2), and the VALUES message does an average across the neighborhood in the original image to produce the reduced value.

Virtual curves and regions can be created. The VALUE message can easily be mapped as described above. A curve can be zoomed by any factor by appropriately modifying the BOUNDARIES and POSITION messages. A virtual curve can also be created that evenly redistributes the points in a source curve in fixed distance increments, or in a fixed number of total points. Multiple curves can be attached end to end by modifying the BOUNDARIES message and dispatching to the appropriate source curve in the POSITION and VALUE messages.

A function to help the user create virtual objects (called MVO) is included in the environment. It has many options, some of which are described below:

1) PMF (Pixel Mapping Function): This option is used for virtual objects that maintain the spatial attributes of their source objects, but modify the returned value. The user supplies <n> source objects and a lambda expression of <n> values. The VALUE message in the newly created virtual object has the effect of applying the lambda expression to corresponding pixels in each of the source objects. For example:

```
(mvo :pmf '(lambda (p1)(logxor -1 p1)) IMAGE)
```

yields

```
:1D-VALUE =
 '(lambda (index)
   (logxor -1
     (send IMAGE :1D-VALUE index)))
```

PMF also understands some keyword options that produce the lambda expressions that are then incorporated into the virtual object. An example of such a usage is:

```
(mvo :pmf '(:scale 1000 2000 250 0) IMAGE)
```

that scales image values 1000 to 2000 into the range 250 to 0. The actual lambda expression that does the scaling is generated inside of MVO.

2) NMF (Neighborhood Mapping Function): This option is used for virtual objects that are neighborhood maps over a single source object. The user supplies a neighborhood size, an optional reduction amount, and a lambda expression of <n> args where <n> is the neighborhood size. The output object is reduced from the source by the reduction factor and applies the neighborhood function to the appropriate region to supply the value in the virtual object. A horizontal gradient example is specified as:

(mvo :NMF '(minus :neighborhood (1 2)) IMAGE).

The neighborhood average example is expressed as

(mvo :NMF '(avg
:neighborhood '(2 2)
:reduction '(2 2)) IMAGE).

3) CONNECT: This option builds a virtual curve that connects any number of source curves as described above. It is created as follows:

(mvo :connect '(:wrap-around t) C1 C2 C3)

The :wrap-around options indicates that the last point should be connected to the first point.

3. PROGRAMMING CONSTRUCTS

Our preferred base language is COMMONLISP because it has the flexibility of any LISP dialect and is a standard language available on a wide variety of hardware. In addition to the base language, constructs are needed for expressing common image understanding idioms in a concise, efficient, and portable manner. Having such programming constructs maximizes the productivity of a researcher by allowing an algorithm to be expressed at a conceptual level without concern for its efficient implementation on a specific machine. The details of generating the efficient code required by image understanding programs can be left to an expert on a particular machine.

Programming constructs for image understanding have two conflicting requirements: to provide as much generality and abstraction as possible while running as efficiently as possible. Many programming environments provide a variety of different constructs to write efficient code on a particular machine. The problem with this approach is that the code is no longer portable and requires full knowledge of the tradeoffs on a particular machine. The approach we have taken is to use macros to create higher level language constructs. These are then compiled into code optimized for the way that a construct is being used and the requirements of a particular machine. This allows the specification of algorithms in a concise and portable fashion, while still hiding the details of efficient implementation.

The two general classes of image understanding programming constructs are object constructs and expansion environments. Object constructs provide well defined ways to create, access and modify objects. They hide the implementation of the objects. Expansion environments provide the context and binding required to make repetitive object operations more efficient.

A typical expansion environment is one that specifies an iteration over neighborhoods in an image, storing the result of applying some function over that neighborhood into another image. The construct provides defaults that can be overridden for figuring the bounds of looping and boundary checking. The image access constructs inside the environment expand into code that 'knows' that accesses are being done in neighborhoods and what sort of boundary checking is required. Other expansion environments provide standard feature interface and error checking or recovery code.

One specific expansion environment that we have developed is called DEFIU. It provides standard keyword arguments, error checking and recovery code, database management code, hooks for interactive function application, boundary checking, and efficient loop generation. It contains knowledge about providing defaults and efficiently implementing common image processing idioms. All of its features are controlled through keywords. It assembles a program from context information and code fragments. DEFIU has proved to be very useful in writing general image processing programs. As an example of the use of DEFIU, see Figure 3-1 showing the definition of chamfer. Each line of this function maps into approximately 20 lines of more primitive lisp code.

```
;;;
;;; Generate a distance map for a binary image by chamfering.
;;;
(defiu chamfer (binary-image)
  "Take a binary image and compute the distance at each point in the
grid from the nearest object."
  menu curve
  input (binary-image)
  output (chamfer-image :array-type 'art-q )

  do (pset (if (= (pget binary-image) 0) 1000000000.0
               0.0)
           chamfer-image)

  in 3 by 3
  centerr 1 centerc 1

  do (pset (min (pget-r chamfer-image 0 0 )
                (+ 2.0 (pget-r chamfer-image 0 -1))
                (+ 3.0 (pget-r chamfer-image -1 -1))
                (+ 2.0 (pget-r chamfer-image -1 0))
                (+ 3.0 (pget-r chamfer-image -1 1)))
           chamfer-image)

  -direction - -direction -

  do (pset (min (pget-r chamfer-image 0 0 )
                (+ 2.0 (pget-r chamfer-image 0 1 ))
                (+ 3.0 (pget-r chamfer-image 1 1 ))
                (+ 2.0 (pget-r chamfer-image 1 0 ))
                (+ 3.0 (pget-r chamfer-image 1 -1)))
           chamfer-image))
```

Figure 3-1: DEFIU

One of the most important of the arguments to a DEFIU function is the Function Application Mask (FAM) keyword. The FAM is a function that can be passed into a DEFIU function to determine whether that function is applied to each individual pixel. This mechanism can be used to restrict the application of a function to a particular part of an image based on the row and column index, a mask image, the pixel's value or any test that can be done with the full flexibility of LISP.

DEFIU provides error checking and recovery facilities to insure that objects are of the appropriate type and that any objects created are destroyed if the function has an error or is aborted. This sort of error recovery is important when writing functions to ensure that the environment is not filled with objects that have corrupted data.

The default boundary checking in DEFIU ensures that no reference will ever be outside the bounds of an object. This default can be overridden on each image individually, so a constant value will be returned by any reference outside the image. Specifying boundary checking once for an object, rather than at every access, has proven to be a very useful feature.

4. SYSTEM DATABASES AND BROWSERS

Several databases have been develolped that automatically record objects and their interactions. This provides a mechanism for storing and finding objects and their relations. Four different databases have been developed: the Perceptual Structures Database (PSDB), the Environmental Database (EDB), the Long Term Database (LTDB), and the Programmers Database (PDB). These databases are implemented as lists of objects, so that additional databases can be easily created by an application. The PDB and the LTDB are maintained in disk files that are automatically read when the system is started. All of these databases are accessible through a common and extensible query language and interactive browsers.

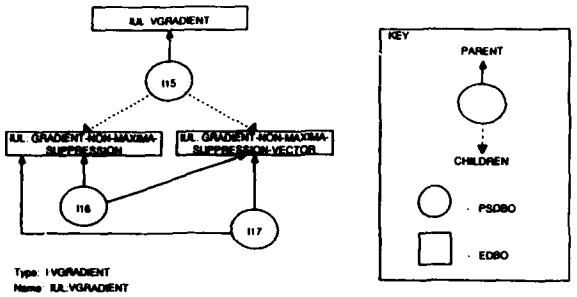
4.1 PERCEPTUAL STRUCTURES DATABASE

The Perceptual Structures Database (PSDB) contains all of the globally accessible perceptual structures generated by an application. The perceptual structures include points, edges and regions, as well as recursive, structure-based groupings of these. The database has two primary purposes: to make an image understanding environment easier to use by providing a central place to find perceptual objects, and to provide a means of communication between different parts of an application. With this database, and tools for exploring it such as filters and browsers, a user can refer to objects by referring to their characteristics as well as their names.

The PSDB contains images and all of the objects extracted from them. Because of its potentially large size, the PSDB is organized hierarchically, with only large-grained objects, such as images, stacks, pyramids, and groups, at the top level. This has two advantages: it combines the information in the database to make it easier to use and it makes queries more efficient by quickly ruling out potential matches for a query. The general combining mechanism is a construct called a *group* that contains a list of objects. A group inherits some of the properties of its contents and can be used with all of the database tools. Each top-level entry in the PSDB has slots for time of creation, parents, children, name and number. The name slot contains the name of a global variable that is bound to that object and the number slot contains a number that refers to that object as well. The parents slot and the children slot are used to make connections to the entries in the EDB. Through these connections, a user can look at the complete processing history of an object.

4.2 ENVIRONMENTAL DATABASE

The Environmental Database (EDB) records function calls, their input parameters, and their results. Each record in the EDB consists of a time stamp, the name of the function applied, the parameters supplied to the function, the input Perceptual Structure Database Objects (PSDBOs), the output PSDBOs, notes included by the function writer, notes added by the function caller, and the name of the person that called the function. When an entry is made in the EDB, that entry is added to the children of each of the input PSDBOs and to the parent of each of the output PSDBOs. In the example Figure 4-1, the EDB records generated by a call first to "vgradient", and then to gradient-non-maxima-suppression are shown. Gradient-non-maxima-suppression vector, called by gradient-non-maxima-suppression-vector, actually generated the returned result. By querying the EDB, all of the PSDBOs generated by vgradient or all records that have a specific note attached to them can be found. Since PSDBOs are linked to records in the EDB, the complete processing history of an object can be shown. This history is useful to track down why an error occurred. The availability of object histories allows both manual and automatic storage management. Since objects in the PSDB are potentially large and cannot be reclaimed by normal LISP garbage collection while they are in a database, it is necessary to be able to remove objects from the PSDB. This creates a problem for the EDB because the pointers to deleted PSDB objects must be removed from function records. This is handled by replacing pointers to a deleted PSDB object with the object's parent EDB record. In this way, space is traded for time. The exact result can be regenerated by using the parent EDB record to call the same function with the same parameters.



Type I VGRADIENT
Name IUL:VGRADIENT
Parameters (#[IMAGE-8 8 6535679] FAM NIL "LEVEL 0 "NOTES "")
Outputs (#[GRADIENT IMAGE-15 15 6651032])
System-Notes "Vgradient calculated in a 2 by 2 neighborhood."
User Notes "Remember that the input image has been smoothed twice."

Type I GRADIENT-NON-MAXIMA-SUPPRESSION
Parameters (#[GRADIENT IMAGE-15 15 6651032] 1.0e-8 USE-GRADIENT-MAGNITUDE NIL NOTES
Outputs (#[IMAGE-16 16 6650992] #[IMAGE-17 17 6651083])

Type I GRADIENT-NON-MAXIMA-SUPPRESSION-VECTOR
Parameters (#[GRADIENT IMAGE-15 15 6651032] 1.0e-8 FAM NIL LEVEL 0 NOTES "")
Outputs (#[IMAGE-16 16 6650992] [#IMAGE-17 17 6651083])

Figure 4-1: Environmental Database Object

4.3 LONG TERM DATABASE

The Long Term Database (LTDB) contains descriptions of all permanent PSDB objects. The LTDB helps when developing an application to manage the large number of objects that are stored. Having a large number of images and preprocessed results available saves time in testing image understanding routines and makes it possible to test a routine on a variety of data sets. Each entry has a name, type, creation-date, source, documentation, directory, files, keywords and history slot.

4.4 PROGRAMMERS DATABASE

The Programmers Database (PDB) automates the management of function and object definitions. It enables a user to browse through the code generated by others to avoid duplicating work. This is particularly important where the environment is being used by many people working on different applications who are not directly communicating with each other.

Automatic maintenance of the PDB is important because users might not spend the effort needed to maintain it manually. Entries in the PDB are automatically updated each time a file is compiled. It is also possible to generate documentation from the documentation in files.

Entries in the PDB have slots for the type, name, parameters, documentation, file, author, and date of last modification. Figure 4-2 shows the PDB entry for the vgradient function. Using the information in these fields, it is easy to use the text browser to find code with specific properties.

#<PDB IUL:VGRADIENT>
Type: DEFIU
Name: IUL:VGRADIENT
Parameters: (IMG)
Documentation: "Calculate the gradient at each point in an image using 2 by 2 neighborhoods."
Author: CHRIS
Date: 12/02/86 16:23:12
File: IUL: RMOD; IMAGEFUNS.BIN.NEWEST
Keywords: (UTILITY "Calculate the gradient at each point in an image using 2 by 2 neighborhoods." CHRIS IUL: RMOD; IMAGEFUNS.BIN.NEWEST IUL:VGRADIENT

Figure 4-2: Programmers Database Object

4.5 FILTER FUNCTIONS

A query language is used to access the databases. The language is designed with the same kind of modularity and combination that is found in Unix pipes, but with more flexibility. The language is based on the combination of filter functions into filters.

A filter is a macro that generates LISP from filter functions and the logical combiners AND, OR, NOT. A filter function takes a list as its first parameter, and, together with optional additional parameters, returns a list as its result. The logical combiners specify how the results of filter functions are piped into other filter functions. This includes the generation of temporary bindings and set union (OR), or difference (NOT) code. Filters are efficient because they expand into some optimal, but harder to understand LISP code. They are very flexible because the only convention is that the first parameter and the results are lists. There are no constraints on what the elements of list are, although they are usually objects. What happens in a filter function can range from simple attribute checking to a complex search or pattern matching operation. The objects returned are usually a subset of the original objects, but can be a superset or even a completely different set of objects. There are three different general classes of filter functions: selectors, transformers, and modifiers.

"Selectors" produce a subset of the original list as their result. This is the most common type of filter function. "Transformers" take in a list and produce a list of completely different objects. "Modifiers" change some aspect of the objects and then return the objects as their result. Side effects in a query can be very useful. For example, long edges can be chosen, their orientation computed and stored, and then only the horizontal edges selected.

Because elements in the lists are usually objects, there are a number of generic filter functions in the core system. Some of these are rangef, memberf, stringf, and timef. "Rangef" selects objects that have a slot value between two other values (or equal to the first if there is only one parameter). "Memberf" selects objects with a slot value that is a member of its second parameter. "Stringf" selects objects that have its second parameter as a substring of the objects slot value. "Timef" selects objects that have a slot value between times. "Collectf" is a transformer that collects all of the objects slot values into a list and then returns the list. There are also a number of special purpose filter functions for specific applications. These functions can be combined and used in many different situations.

Some example filters include:

```
(filter *pdb*
        (memberf :author '(chris phil))
        (timef :date "Jan 1")
        (or (stringf :keywords "edge")
            (stringf :keywords "curve")))
```

Return all entries in the Programmers database that were written by chris or phil after January 1 of this year that have something to do with edges or curves.

```
(filter 13
        (near-object *curve* 10)
        (within-filter :avg-intensity 3))
```

Return all curves in group 13 (an application specific database) that are within 10 pixels of the curves position in its label plane and are within 3 of the objects average intensity.

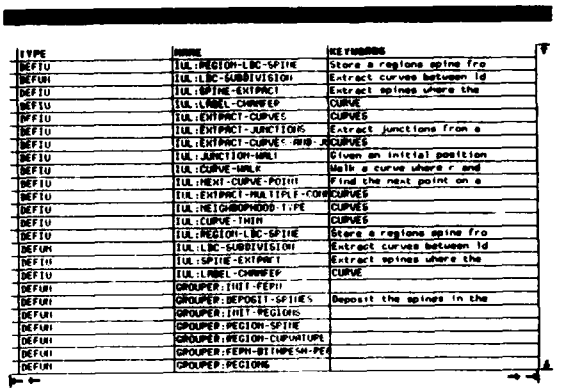


Figure 4-3: Text Browser

4.6 BROWSERS

Browsers are an interactive query mechanism for finding and exploring relationships between objects. Three different types of browsers have been designed and partially implemented: a general text-oriented browser, an image browser and a graph browser. All of the browsers maximize the amount of information breadth or depth presented at one time and provide a means to interactively apply functions to selected objects. They follow the filter function convention of using lists as their first parameter and of producing lists as results, so they can be used in the midst of a filter.

4.6.1 Text Browser

The text-oriented browser works with any object and provides an easy interface to interactively apply functions. There are two different display modes. The "object/line mode" presents each object on a single line of text divided into fields and maximizes the breadth of information that can be seen at one time. The "field line" mode displays one object at a time with one field per line, maximizing the depth of information displayed about that one object. The object/line display is useful when looking for a specific set of objects. As the set of objects is narrowed down, the set of objects that meet selected specifications are shown. The field/line display is useful when stepping through a subset of the objects in a database, looking at each object in detail. Figure 4-3 shows the text browser being applied to the PDB.

4.6.2 Image Browser

The image browser displays reduced resolution pictures of objects. Usually a text browser or a filter is used to narrow down the number of images to browse. The image browser can be used to scroll through the pictures, look at their entries in the LTDB and load the objects. The pictures are generated automatically on a request to browse a specific object and then stored so that if that object is browsed again, the browser can use the previously generated picture.

4.6.3 Graph Browser

The graph browser displays objects and the relationships between them. In the same way that the image browser shows what a stored object looks like more clearly than its textual description, the graph browser shows the connections between objects more clearly. It can be used to look at the relationships between objects in the PSDB and EDB, or to trace the processing flow through object models.

5. DISPLAY SUBSYSTEM

The display system is built around the family of messages described in the section on object access. Each routine contains two main pieces; an iteration scheme for scanning and accessing the object using the basic messages, and the code needed to display the object on the screen. Because all objects can be accessed the same way, a small set of display functions can be used on a wide variety of objects. The use of virtual objects enhances and simplifies the display system. Rather than having options to scale, reduce, zoom (etc) images in the display routine, a virtual object can be created that captures the desired functionality. The display function can concentrate simply on presenting an object.

The user interacts with the display system through a single function (the "D" function) as shown in Figure 5-1. This function plays two roles. It handles many of the tedious details associated with window management, and it provides a uniform syntax for issuing display commands. The current pan, zoom, offset, sub-image and sampling rate for each window are stored in an internal database and can be used as the default in subsequent displays (for displays that support these features). Windows can be assigned as viewports on other windows and are automatically

In general, a display is colored based on the value of the underlying object. However, this default can be overridden by over-coloring, another display option. For example, a mountain of elevation 200 can be displayed in color 200 in a standard call to the surface display. However, it is often desirable to guide a display with one set of objects, and color it with another. A surface display may use a terrain image as the source of elevation data, and a feature image to color each point. Another example is to use a horizontal gradient and vertical gradient image to display a vector field but use the absolute magnitude of the pixel underlying the vector as the color. Both these tasks are straightforward using the over-color. Over-colors, similar to FAMs, are objects in direct correspondence with an object being displayed. Rather than serving as a binary mask, it is used as the color indicator. Over-colors can also be fixed values (as opposed to objects) if a monochrome display is desired.

The following display routines are available in the initial environment.

- Pixel: Basic display.
- Edge: A pixel and its neighbors are submitted to an edge function, and, if this function returns non-zero or non-nil, the pixel is plotted.
- Vector: Expects two images, the first specifies the horizontal component of each vector and the second is the vertical component. Vectors are based on imaginary grid lines through the image. The grid position and size can be specified by the user. FAMs can also be used to mask off the grid.
- Surface: Perspective display of an image interpreted as elevation data. Appears as a warped grid of lines. Position, elevation above the ground, pan and tilt angle, maximum visibility, field of view, and resolution can all be specified. This routine optionally builds a visibility mask: a binary array where pixels visible from the current orientation are indicated.
- Paint: Surface display with shading. This routine optionally stores the inverse transform from perspective coordinates back to image coordinates for every point on the viewing surface.
- Curve: Iterates as follows over its arguments: For all 1D indices, at the 2D POSITION, plot the 1D VALUE, connecting adjacent points. Like the region display below, this routine is optimized for particular objects, namely 1D curves. Note that an image is also displayable with this routine.
- Region: Iterates as follows over its arguments: For all rows and all columns in the 2D BOUNDARIES, plot the 2D VALUE at row,column. This routine is optimized for region objects stored as chain codes but also functions correctly (if slowly) on curves and images.

```
1) Display Zero-Crossings in Red:

(d :edge (mvo :PMF '- (smooth-n image 2) (smooth-n image 4))
  :edge-function '(lambda (p1 p2) (not (= (sign p1)(sign p2))))
  :over-color    red)

2) Contour plot every *contour* levels:

(d :edge image :edge-function '(lambda (p1 p2)
                                 (not (= (// p1 *contour*)
                                         (// p2 *contour*)))))

3) Vector display of the gradient colored by the magnitude:

(d :vector (mvo :nmf (#'- :neighborhood '(1 2) :reduction '(1 1)) image)
           (mvo :nmf (#'- :neighborhood '(2 1) :reduction '(1 1)) image)
  :over-color image)

4) Display a virtual closed curve composed of 3 connected smaller curves

(d :curve (mvo :connect (:wrap-around t) curve1 curve2 curve3))
```

Figure 5-1: Display Examples

updated as the main window is changed. All display commands are stored and can be reexecuted and cycled on a window by window basis. Each window can be assigned a set of colors (or dashed line patterns on a monochrome display) that can be used or cycled in particular displays. The interface function also checks for errors in the calling parameters of a particular display routine before the function is called, and can route a display request to an optimized routine when appropriate.

The D function also simplifies the implementation of display routines. It can do some of the error checking normally associated with a display function. The interface manages the task of defaulting display parameters for each window, so the programmer can depend on his display routine being called with actual, valid values. The interface is responsible for queueing display commands and handling viewports, removing the burden from the actual display routine. The researcher can use any of the above features when installing his new display routine by characterizing his program in any of the several internal databases that handle display requests.

Displays can be modified to highlight selected attributes through the use of Function Application Masks (FAMs). A function application mask (FAM) is an object in direct correspondence with the object being displayed. It serves as binary mask that determines whether or not a particular data point should be displayed. For example, in a 3D terrain display, a FAM can be used to flatten a group of mountains and see what's behind them by inhibiting the display of any pixels corresponding to the range. Another example is to use a FAM in a vector display so that only vectors of a certain magnitude are allowed. Note that in this case, the FAM object is actually more complicated than the data objects (which supply the delta-X and delta-Y values) because it must reference both data sources and threshold their cartesian length. Note that a FAM is not needed for the first example - a virtual object can be created that incorporates both the original terrain elevation data and the terrain feature data, and zeros the elevation wherever a mountain is indicated - but is required in the second example where no vector (as opposed to a zero length vector) is desired.

There is also a routine called display-values that allows a user to interactively select points in multiple objects and apply arbitrary functions to the returned values. With this, a user can browse "inside" an object description. It also allows the user to manually supervise localized processing. The user can also modify the underlying objects based on data he can interactively indicate in this routine. Display-values is especially useful when applied to label planes - functions can be applied to user selected objects. A routine for interactively segmenting an image into point, curve and region objects has been built on top of this function.

6. SUMMARY

The variety and flexibility of this computer vision development environment allows us to quickly implement and experiment with new algorithms and approaches. The combination of a set of basic extendible objects with uniform interfaces, virtual objects, system databases, interactive browsers and a powerful display system is a synergistic one. Each piece amplifies the usefulness of the others. The environment has been successfully developed and used by researchers working on a variety of problems. They have found the overall environment easy to use and powerful, although there are still areas that need additional development. Most of the constructs described in this paper have been implemented for the Symbolics LISP machine using ZetaLisp and Flavors. The system is currently being ported to a Commonlisp environment.

ACKNOWLEDGEMENTS

The authors wish to thank John Dye, Kaiti Riley, Danny Edelson, Jay Glicksman, Angela Erickson, and Laura McConnell.

This work was largely supported under U.S. Government contract number DACA76-85-C-0005 for the U.S. Army Engineer Topographic Laboratories (ETL), Fort Belvoir, Virginia, and the Defense Advanced Research Projects Agency (DARPA), Arlington, Virginia.

REFERENCES

Donaldson et.al. - 83] - D. Donaldson, R. Kirby, J. Pallas, and A. Rosenfeld, "UMIPS: University of Maryland Image Processing Software", Computer Vision Laboratory, Computer Science Center, Maryland, December, 1983.

Kohler et.al. - 83] - R. Kohler, et.al., "VISIONS: Operating System Reference Manual", Computer and Information Science Dept., University of Massachusetts at Amherst, 1983.

Quam - 84 - L. Quam, "IMAGECALC: Reference Manual", Stanford Research Institute, Menlo Park, California, 1984.

INTERPRETING AERIAL PHOTOGRAPHS BY SEGMENTATION AND SEARCH

David Harwood
Susan Chang
Larry S. Davis

Center for Automation Research
University of Maryland
College Park, MD 20742

ABSTRACT

A knowledge-based system for interpreting aerial photographs, Picture Query (PQ), first segments an image into primitive, homogeneous regions, then searches among combinations of these to find instances which satisfy definitions of object types. If primary evidence is insufficient, there may be an hypothesis-based search for the supporting evidence of related objects. This secondary search is restricted to windows by expected spatial relations. First instances are improved by searching for overlapping variants having better goodness-of-figure. The process may be repeated using re-estimated parameters of object definitions based on instances found previously. Results are reported for images of suburban neighborhoods, including roads, houses, and their shadows.

1. INTRODUCTION

Analysis and interpretation of images are the principal objectives of computer vision. This report describes a system, called Picture Query (PQ), for two-dimensional interpretation of images and its application to aerial photos of suburban neighborhoods.

There are many image segmentation methods based on intrinsic, usually local, image properties, such as reflectance, texture, or adjacency. Some use simple parameters or rules derived from semantical or physical considerations about very restricted scenes, in addition to heuristics about general imagery. However, this information is inadequate for defining even common objects; most of these methods do not model important geometric properties. Consequently, such segmentation methods do not reliably segment regions corresponding to important objects, and their behavior is difficult to characterize.

Our system defines types of and relations among objects, and uses these definitions to test for instances. It also uses an initial, hierarchical segmentation and a heuristic search to incrementally segment the image into specified objects by merging. Because of the use of general definitions, incremental instantiation, and complete search, the results are reliable, and the procedure is easy to characterize.

PQ consists of three subsystems: a high-level vision system (HLVS) representing goals or queries and knowledge about objects, a low-level vision system (LLVS) for image segmentation and evaluation of predicates, and a graph search procedure to find instances by merging and testing.

The LLVS segments the image into a one-parameter family of segmentations containing increasingly refined connected components. For each object type, a suitable segmentation is used as a basis for a heuristic search to find instances, by merging combinations of components and testing their properties and relations. The HLVS communicates with the user about goals and about definitions of objects by image predicates, maintains a database of current interpretations, and provides segmentation, search, and optimization procedures with parameters associated with object types.

Important characteristics of PQ's data representation are description of object types by general image predicates, including geometric ones, which are evaluated logically, and representation of images by attributed adjacency graphs associated with variably refined segmentations into connected components. Other important characteristics of the control structure are reinterpretation with adjusted parameters, optimization of instances for goodness-of-figure, and reasoning with expected spatial relations. These will be explained further in appropriate chapters.

2. HIGH-LEVEL VISION SYSTEM

The High-Level Vision System (HLVS) interactively communicates with the user concerning queries, parameters, and definitions of objects to be found in images. It represents descriptions of types of objects, as well as associated parameters which are used during segmentation, search, and optimization. In addition, it maintains a database of facts giving the current state of interpretation. We may consider this system to be the interpreter of the image data, but especially of the uninterpreted, primitive segmentation given by the Low-Level Vision System (LLVS); this interpretation proceeds by heuristic search for instances of object types. The LLVS and the search procedure will be described in the next two chapters.

In a complex scene, segmentation by a fixed procedure usually does not allow simple identification of important parts of an image. Some objects may appear to be fragmented with any choice of parameters, so that merging may be necessary; also, some segmentations are more suitable for finding some object types than others. In addition, interpretation may depend on more complicated reasoning involving the existence

and locations of secondary objects. Finally, having identified some instances of objects, it may be necessary to adjust some parameter values so that the remaining instances may be discovered.

We now describe the types of knowledge used by the HLVS to control interpretation, including modeling of objects by definitions with parameters, reinterpretation with adjusted parameters, optimization of instances according to goodness of figure, and contingent search using spatial relations.

2.1. Object modeling

The models of objects are descriptions of their properties and relations as they appear in images. Here we are concerned with two-dimensional interpretation of images, so that we will identify object types with types of images, defining them by image predicates, for example, by grey-value statistics, areas and distances, and tangent directions of boundary chains.

Ordinarily, an instance of an object type is a single connected component, although it may also consist of multiple components. Spatial relations are defined between same or different object types, for example, between adjacent houses or between houses and their shadows.

The description of an object type consists of properties of and relations between connected components or their boundaries. In our system, these include adjacency and relative position, average grey-value, area, compactness, boundary length, and rectilinearity.

Components are defined by 8-connectivity. Compactness is defined by the ratio of area to squared perimeter, and is used as a goodness-of-figure measure for houses since it decreases when holes and gaps are present. A component is said to be rectilinear if more than half of the tangents to its boundary lie in some pair of orthogonal directions.

The definitions of object types by predicates have the form of conjunctions of inequalities involving property functions. For example, house(x) = (min$<$ area(x) $<$max) and (min$<$ brightness(x) $<$max) and (min$<$ compactness(x)) and (min$<$ rectilinearity(x)), where there are several parameter bounds. In our present system, we only define road complexes, that is, connected complexes of streets and driveways which are not analyzed further. Shadows and houses are defined with the same property functions, while road complexes are not tested for area or rectilinearity; different parameters are associated with different object types.

Relative position is defined by a window of given size and a position specified relative to some known object in the image. It is used to restrict contingent searches for secondary objects which might provide supporting evidence or whose existence is otherwise expected. For example, if the evidence for a house is insufficient, supporting evidence of its shadow might be sought. Also, if a row of houses has been found, a search might disclose intermediate instances where there are large gaps in the row.

2.2. Parameter adjustment

Parameters of object descriptions are usually derived from prior expert opinion about the given domain. However, these may not be very useful in general because of variations due to imaging. For example, roofs may appear to be brownish rather than silver-white, due to inclination of the sun, or to regional differences. As a result, some instances of an object may be found, while others are not.

Our system attempts to confirm whether the parameters of the model are correct, comparing the means of properties of found instances to current model parameter values. If the difference is large, the image is reinterpreted with respect to an adjusted object description, using the means as new parameter values, but with tighter bounds than at first. This readjustment of parameters may be repeated until the results are unchanged.

2.3. Optimization of instances

When an acceptable instance of an object type is first found by merging primitive components, it may be that there are variant regions, overlapping the one first found, which are more acceptable as instances. For example, there may be obvious gaps or irregularities in boundaries, or interior holes.

We want to find the variant of the first-found instance which has best goodness-of-figure for the given type of object. The reason for this is that the graph search procedure may stop with different results depending on its starting point (seed component) and its rules for expanding search. Because definitions must be general, so that tests do not discriminate against variable but true instances, the search may not stop at the best candidate among the set of acceptable variants.

In the case of man-made objects, such as houses and roads, as well as many natural objects, such as trees, instances not only have significant figure-ground contrast, but usually have regular or smooth boundaries and filled interiors. Object definitions usually require homogeneity of reflectance properties, guaranteeing figure-ground contrast. It is natural, then, to use compactness as a measure of goodness-of-figure to select among variant instances.

The optimization procedure is simple: starting with the first-found instance, find the variant differing by at most one primitive component, which is also acceptable, and having the greatest compactness. This is repeated until no better candidate is found, with the final best one being returned as the instance found by the search after optimization. This simple heuristic procedure has proved to be very effective in most cases, but it should be possible to define similar measures of goodness-of-figure for different types of objects.

2.4. Reasoning with spatial relations

Humans often use expected spatial relations among objects in a scene to find subtle or obscure instances whose recognition is contingent upon having already found instances of related objects. For example, identifying or hypothesizing elevated objects, in conjunction with physical knowledge, makes it possible to recognize shadows or occluded objects.

There are dual aspects of reasoning involving spatial relations: prediction and confirmation. On the one hand, having found an object, we may expect to find spatially related objects by a restricted search; on the other hand, finding the related objects may tend to confirm our identification of the first object, which may have been tentative or hypothetical. Relations usually represent mutual information, even if they do not definitively identify either of the related objects. Thus we may use expected spatial relations to locate missing objects or to confirm or disconfirm uncertain hypotheses.

2.4.1. Missing objects

Sometimes instances of objects, or partial instances, are missed because of their unusual appearance or because they

are obscured. For example, houses or roads may be partly overshadowed. Still, we may expect them because of our knowledge about regular configurations of houses along roads.

Our system attempts to recover missing houses using expected relative positions between instances. A less restrictive correct position relative to a previously found instance but which did not itself yield an acceptable instance of a house during search by merging.

2.4.2. Supporting evidence

There may be considerable variation of appearance in instances of a given object type, so that it may be necessary to relax the parameters of the definitions in order to find instances whose evidence is weak. Furthermore, in these marginal cases, our system may consider evidence arising from any types of objects which are expected to be spatially related to an uncertain candidate. For each such type, a search is made in a window positioned relative to the uncertain candidate. If some instance of a related object is found, then identification of the first candidate is confirmed; otherwise it is rejected.

This concludes our description of the HLVS which represents and controls interpretation of images. We now consider the LLVS which produces variably refined segmentations of the image, along with attributed adjacency graphs, and then consider the search procedure which generates and tests merged regions to find instances of object types.

3. LOW-LEVEL VISION SYSTEM

The LLVS system of PQ is quite simple, consisting of image enhancement by iteration of an edge-preserving smoothing operator, computation of a one-parameter family of increasingly refined segmentations of the enhanced image into homogeneous connected components, and a library of functions used in defining object types by predicates and in testing candidate instances during search.

3.1. Enhancing the image

The input image is enhanced by iteratively applying the symmetric-nearest-neighbor (SNN) operator, sharpening edges while flattening interiors of regions [Harwood, 1984]. This operator is very efficient and converges rapidly without introducing artifacts; there are also versions for vector images. About 20 iterations are usually necessary to achieve a stable, enhanced image, one that is 99.9 percent fixed.

3.2. Connected components

The enhanced image is segmented into homogeneous, or almost-constant, connected components by a standard two-pass routine for computing 8-connected components, in which adjacent pixels are regarded as connected if the difference of their feature values, here grey-value alone, is sufficiently small, i.e., less than the threshold parameter of the segmentation. The family of increasingly refined segmentations is parameterized by this threshold value, with higher parameter values resulting in segmentations into fewer, larger, higher-contrast components, and lower values resulting in many smaller, lower-contrast components.

3.3. Attributed adjacency graphs

Associated with each segmentation is an attributed graph representing the adjacency relation and some properties (area, mean grey-value) of the primitive components. The family of segmentations, together with their graphs, is the basis for the search procedure, whose goal is to find instances of object types by merging components and testing their properties and relations.

3.4. Image predicates

During search, various functions of candidate regions or of their boundaries are repeatedly computed, including area, grey-value, perimeter, tangent directions of boundary, so that properties and relations of candidates may be tested as to whether they satisfy the definitions of object types.

4. SEARCH FOR INSTANCES

The search procedure is given a query, or goal, to return the first-found instance of an object type, found by merging and testing regions within a specified window. The window may be the entire image, or may be determined by expected spatial relations among objects. All instances of an object type may be found by successive searches of the remainder of the window.

4.1. Coarse-to-fine strategy

Search first uses a coarse segmentation, chosen according to prior knowledge of the expected size and contrast of the object type in the query. If necessary, search may continue with more expensive computations based on refined segmentations, until an instance is found, or until the search has failed at every level. This coarse-to-fine strategy, beginning with an initial segmentation appropriate to the object type, is efficient for finding instances when they exist.

4.2. Seeds

Search starts from seed components having features which are typical of the object type. Since these seeds are often smaller than the complete instances, they have to be simply characterized by local features, such as grey-value. Local feature vectors, based on color or texture, would be much more useful for characterizing seeds. The seeds may be ordered by difference of their feature values from the typical values (or by likelihood). In our system, every object instance found by merging must originate from some seed component having typical grey-value.

4.3. Merging regions

Our method of search by merging primitive regions is a modified graph search, operating on the adjacency graph of a segmentation; it will be described in the following subsections.

4.3.1. Motivation

It is impossible to define general procedures for segmenting images of natural objects, except when their properties are very uniform, and when object-background contrast is large. Simple procedures like recursive thresholding or split and merge algorithms are fairly efficient and give apparently good segmentations of some images. But the problem with these is not that they generally have wrong parameters, but that they do not really search for identifiable objects or reason about their relations.

There are a number of difficulties in implementing search procedures for interpretation, including adequate description of objects, evaluation of evidence, efficiency and completeness,

and effective heuristics for control. But these difficulties seem to be worth overcoming if the resulting system is generally applicable and achieves robust and sound results, at the cost of time occasionally spent in exhaustive, but futile searches. (There may be good heuristics to cut these off.)

4.3.2. Basic concepts

The search algorithm is bounded-depth, best-first search based on merging and testing of candidate regions, using a heuristic merit function to direct the search. The terms used to describe the search are briefly explained in the following paragraphs.

The adjacency graph consists of a set of nodes representing components, together with edges representing their adjacencies.

Another graph, the search graph, is generated during the search process, representing information about merged regions. A node of this graph represents a merged region and gives its merit value, its size and average grey-value; the merged region is identified by the path of components which were merged, starting with the seed region as root, resulting in the current region.

The successors of a node represent results of merging new, adjacent components with the region represented by the node.

The open list is a priority queue of components which are adjacent to the current merged region and have not yet been considered for expansion. Their order of merit is based on the differences between their average grey-values and the expected value of the object.

The closed list consists of those components which have already been eliminated.

For more details about graph search, see [Rich, 1983].

4.3.3. Algorithm for modified graph search

(1) Create an initial search graph G, consisting solely of the start node S, representing the best seed in the list of seeds. Create empty open and closed lists, then put S on the open list.
(2) Loop: if the open list is empty, quit the search with failure to find an instance of the object type.
(3) Put the first open node on the closed list, calling it N.
(4) Expand node N into successors which satisfy acceptance criteria, as detailed in Section 5.3.4.
(5) For each successor in turn, check whether stopping criteria (detailed in section 5.3.5) are satisfied. If so, quit the search successfully, returning the resulting instance.
(6) Reorder the open list according to merit.
(7) Repeat Loop.

The search procedure is then restarted with a new seed until the search succeeds or the seed list is exhausted.

4.3.4. Acceptance criteria

Let M be the region represented by the node which is currently to be expanded. Let A be a component to be tested for acceptability, and R be the result of merging M and A. Then A is acceptable if it satisfies all the following conditions:
(1) A and M are connected.
(2) R is different from every closed node; this avoids redundant search and calculation, at the lesser cost of tabulation and checking.
(3) The area of R is less than the maximum area of the object, and its average brightness is within bounds.
(4) One of the following rules for merging, based on similarity and size, must be satisfied:
Rule (i): Regardless of their sizes, merge A and M if their feature differences are small enough.
Rule (ii): Otherwise, if A is large enough, merge A and M if their feature differences are small enough under a relaxed constraint. This rule allows merging of larger, somewhat dissimilar regions, as long as condition (3) warrants their merging. This may apply to surfaces with varying orientation, e.g., roofs of houses.
Rule (iii): If the area of A is very small, merge it with M. A may be an artifact or noise.

Generally, other rules of limited applicability may be needed to handle special situations. For other merging rules, see [Nazif, 1984].

Preliminary tests of model properties are contained in condition (3). As the graph search proceeds, merged regions become too large to be expanded. The other conditions restrict merging so that it is non-redundant and preserves homogeneity and connectivity.

4.3.5. Stopping criteria

The search stops with failure when it exceeds the bound on depth without finding an acceptable instance.

On the other hand, when merging results in finding a candidate region which is an instance of the object type, meeting all the criteria of its definition, the search might stop successfully, returning the instance. We will call this version FF search, and the returned instance will be called the first-found instance.

The problem with this procedure is that the first-found instance may have gaps at its boundaries or holes in its interiors, so that even though it passes other tests, it is not a very good instance. Moreover, there may be overlapping variants of the first-found instance which are better instances. Generally, the first-found instance will be too small, being incomplete.

It would be very expensive to search the entire space of acceptable candidates, trying to find one which is the best instance. A different approach is to expand according to the merit function until some instance is found which cannot be further acceptably expanded. This version will be called FL search, and the returned instance will be called the first-large instance. A comparison of the two approaches will be presented in Chapter 6.

5. EXAMPLES

This chapter presents some representative examples of the application of PQ to the analysis and interpretation of aerial photos of suburban neighborhoods. The search for instances of roads, houses, and shadows of houses is described in detail.

5.1. Goal

The goal of PQ is to find connected regions, obtained by merging, which are instances of object types. In the following examples, we consider houses and their shadows, as well as

roads, but other types of objects might also be considered by adding their definitions. The system may return from search with all disjoint instances, or with the first-found, possibly optimized instance of the requested type, if any.

The system asks the user to specify what type of object is to be found in what window of the image—for example, houses in the entire image. There may be several such goals on a stack, which are considered in order.

Given a current goal, PQ first checks its database to find whether some instance(s) have already been found which satisfy the goal. If not, PQ searches for instances as specified by the goal.

5.2. Enhancement and initial segmentation

Iterative application of the SNN operator produces an enhanced image having flat regions with fixed, sharpened edges. Figure 1a shows the original 440x160 image of a suburban neighborhood, and Figure 1b shows the enhanced image.

The initial segmentation of the enhanced image is obtained by a connected components algorithm, in which adjacent pixels are connected if their grey-value difference is less than a threshold parameter value. Finer segmentations are associated with lower threshold values, and values are chosen for searches according to the expected sizes and contrasts of object types.

Figure 2 shows a 44x40 enhanced image of a house and yard and its segmentation into connected components.

In general, the number of components depends on the threshold value, and also on image size and contrast. Figure 3 shows high- and low-contrast segmentations of an image of a neighborhood using two thresholds; the image with lower contrast has fewer and larger components.

5.3. Search for instances

Given a query about a type of object, PQ searches for instances by merging and testing regions. The search originates at seed components having typical grey-values and continues until a first-found or first-large instance is found, or until all candidates fail or search exceeds the depth bound.

Figure 4 illustrates how seed components for houses are selected from from the components of a house and yard, also shown in Figure 2. Here seed components 1,4,5, and 6 were selected by constraints on their grey-value ($110 < g < 255$) and area (<310), and then ordered 6,5,4,1 according to the difference between their grey-value and the typical value.

5.3.1. Version FF graph search

In our example, the search graph initially consists of a single node representing seed 6. Its area and average grey-value are 94 and 152, as shown in the table in Figure 4c.

In the search graphs shown in Figure 5 and 6, nodes that are closed, having been considered already, are dark circles, while nodes that are still open are white circles. A new cycle of merging and testing begins whenever an open node is closed.

In the first cycle, region-merging starts with seed component 6. The table in Figure 4 shows that components 0,4, and 5 are adjacent to component 6, so each of these in turn will be checked. Component 0 fails to satisfy condition (4) on area and grey-value, but component 4 is acceptable by all criteria, so it is chosen to be merged in the successor of node (6), which is checked against the stopping criteria. The search then successfully terminates, since the merged region consisting of components 6 and 4 satisfies the definition of a house. This region is entered in the database as an instance of a house, while its individual components are removed from further search.

Figure 5 shows the search graph of the last two cycles, and Figure 5c shows the instance found. This instance has a hole caused by early termination of the graph search; component 5 which fills the hole is ignored. The remaining seeds, 0 and 5, fail to grow into acceptable house candidates since components 6 and 4 have been removed from further search.

5.3.2. Version FL graph search

The Version FF search terminates with a first-found instance, even though there may be holes or gaps in the resulting figure, as in the above example. Version FL search, on the other hand, terminates with a first-large instance which cannot be further expanded.

The same image may be used to compare the two versions. As before, seed 6 is selected first, and its adjacent components are 0,4, and 5. While 0 remains unacceptable, 4 and 5 are acceptable for expanding node (6). In the second cycle, the node (6,5), having greater merit, is chosen for expansion; it has unused adjacent components 0 and 4. As before, component 0 fails condition (4). Component 4 is acceptable, and so is merged in the successor in the path. In the third cycle, the current node (6,5,4) has adjacent components 0 and 1, of which 1 is acceptable. Finally, the node (6,5,4,1) cannot be acceptably expanded.

This merged region (the union of components 6,5,4, and 1) is now tested and satisfies the definition of a house, so is returned as the first-large instance. If a candidate does not satisfy the definition, the search continues to expand open nodes to find a first-large instance. Figure 6 shows the FL graph search for this example, and the returned first-large instance.

5.4. Reasoning by the HLVS

The HLVS uses its database of accumulated facts about found instances to control parameter adjustment, optimization of instances, and contingent search using spatial relations.

5.4.1. Adjusting model parameters

The HLV system compares mean values of properties of found instances with current expected values. When the differences are large, the current values are replaced by the sample means, and the image is reinterpreted with respect to the new expected values. This procedure may be repeated until the results are nearly unchanged.

To illustrate this, Figure 7 shows the result of interpreting a 480x410 image before and after adjusting the area and grey-value parameters used to define houses. After one step, the segmentation is obviously improved, and after three iterations, the parameter values are nearly unchanged. The same number (47) of true instances have been found, but seven of ten false instances have been eliminated.

5.4.2. Optimizing instances

Figure 5 shows a hole in a a first-found instance of a house. Its compactness, the ratio of area and squared perimeter (a measure of goodness-of-figure), is 0.033. Among all variants which differ from this first instance by at most one com-

ponent, the candidate consisting of components 4,5, and 6 is also an acceptable instance, but with an improved value of compactness, 0.067.

Iteration of this procedure continues until compactness can no longer be improved. In the previous example, a second iteration does not find a more compact instance, so that the instance consisting of components 4,5, and 6 is considered optimal. Figure 8 shows the result of optimization for this example, and also for another more complex example. Incremental optimization of instances with respect to compactness is usually effective, although it may be possible to achieve good results for other types of objects with different measures of goodness-of-figure, perhaps based on contrast at boundaries.

5.4.3. Contingent search

Contingent search for spatially-related types of objects, based on identifying or hypothesizing a given instance, may be used to find missing or predicted objects, or to confirm or reject tentative hypotheses about the first object.

5.4.3.1. Supporting evidence

The system may find additional instances of object types within low-contrast or complicated images by relaxing object definitions. Given such an uncertain object hypothesis, the system may search for instances of related objects that would tend to support or disconfirm the hypothesis.

Figure 9 shows six candidate houses in an image, of which the two in the upper right are uncertain instances, that are hypothesized only when the parameters of the definition have been relaxed. To confirm these instances, the HLVS initiates searches within restricted windows for the shadows of the two candidate houses. The result of this search is shown in Figure 9c.

5.4.3.2. Finding missing objects

Sometimes additional instances of object types are expected on the basis of previous identification of related instances. For example, Figure 10 shows that the fourth house in the top row is overshadowed so that it is not found; also, partial instances in the margin of the image are missed. Since neighboring instances have been identified, the system initiates searches within windows determined by the neighboring instances, beginning with seed components which previously failed to grow into instances. Because of the expected spatial relations, the system provisionally relaxes the definition of a house to admit, as an instance of part of a house, any seed or acceptable expansion within the search window. When this is done, Figure 10c shows that three missing small parts of houses are found.

5.5. Other experimental results

Figures 11-14 show images of four different suburban neighborhoods, their initial segmentations, and instances of houses and roads obtained by merging and testing. All these experiments involve FL search, reinterpretation with parameter adjustment, and use of contingent searches for missing houses. Some illustrate optimization as well.

Merging to find roads is difficult, requiring special treatment, since they are elongated, adjacent to many components along a road, and sometimes have poor figure-ground contrast. Given these difficulties, only when primitive regions are sufficiently large are they acceptable for merging as road complexes.

Figure 11 shows the instances of houses found by FL search, without optimization, starting with a coarse, high-contrast segmentation into components. It also shows the instances of a road complex found in a finer, lower-contrast segmentation.

Figure 12 gives similar results for a more complicated image. The results are still very good, correctly identifying all but perhaps two of fifty-one instances of houses, including two difficult overshadowed instances, with only two false instances. All important roads were also found in this image, which has good contrast but lower resolution.

Figures 13 and 14 show optimization of the FL instances of houses. The optimized instances generally have better shapes, with fewer gaps and protruding components.

6. RELATED WORK

The goal of the PQ system is to find regions in an image which correspond to certain objects in a scene. Developing such a model-guided segmentation system involves addressing several issues.

The first issue is image representation. [Marr, 1982] suggested that an edge-based primal sketch would capture most of the salient information about an image. In a similar spirit, Feldman and Yakimovsky [Feldman, 1974] segmented images into primitive regions of almost constant brightness. Since contrast between pixels is used to determine these regions, this representation preserves much of the edge information of Marr's primal sketch.

The second issue is how to apply knowledge in segmentation. Some image understanding (IU) systems restrict the knowledge to the high-level portion of the system, using it to check the results from the low-level portion. Such systems often use simple segmentation methods such as thresholding. See [Selfrige, 1982], [Hwang, 1985]. Other IU systems pass knowledge in the form of parameters or optimization criteria to their low-level components, which then use segmentation methods involving search; see [Feldman, 1974] and [McKeown, 1984].

In the following subsections, five representative systems related to PQ are discussed.

6.1. Feldman and Yakimovsky

In [Feldman, 1974], a combination of mathematical decision theory and heuristic search techniques is used to develop a general-purpose system for scene analysis.

A preliminary segmentation procedure partitions the image into primitive regions. Each primitive region is initially assigned various probabilities of interpretation, based on the values of measurements (e.g. color, area), using knowledge about possible objects in the scene.

After this preliminary evaluation, the system uses a heuristic search to merge primitive regions, as does PQ. The goals of the two systems are different. The goal of the former system is to find the best global interpretation, the one having the greatest likelihood, while PQ's goal is partial interpretation by extracting regions that satisfy object descriptions. The merit function, acceptance criteria, and stopping criteria used in the search are very different.

6.2. Barrow and Tenenbaum

In IGS [Tenenbaum, 1977], knowledge from a variety of

sources is used to make inferences about the interpretations of regions. A scene is first partitioned into elementary regions consisting of individual pixels or groups of adjacent pixels with identical attributes. Beginning with this partition, IGS first performs the most complete interpretation possible in the current partition. Based on this interpretation, it next merges a pair of adjacent regions that are least likely to represent distinct objects. Merging is repeated until all adjacent regions have different interpretations.

In a later, different approach, called MSYS [Barrow, 1976], regions are assigned sets of possible interpretations with associated likelihoods based on local attributes (e.g., color, size, and shape). A relaxation process is then applied repeatedly to adjust the likelihoods up or down based on the interpretation likelihoods of related regions, until a consistent set of likelihood values is attained. At this stage, several alternative interpretations may still exist for some regions. The final stage of analysis involves searching for a set of unique interpretations with the highest joint likelihood.

6.3. McKeown and Enlinger

MACHINESEG [McKeown, 1984] uses map knowledge to guide image segmentation. The map-to-image correspondence can be derived from camera and terrain models. The location of each map feature in the map database can then be projected onto the image. Map knowledge is used to determine criteria for region-growing.

The merge algorithm is straightforward. Initially, connected components of homogeneous image intensity are produced. A list of the edges between regions is sorted by the strengths of the edges. Criteria for testing merges are derived from map-based descriptions. Regions separated by weak edges are merged first. Each time a new region is created, it is checked against the criteria to see if it can be identified as a prototype region. The processing can be repeated until some merged region fits the criteria, or until process exceeds a given depth.

In contrast with MACHINESEG, PQ uses variably fine segmentations rather than using fixed primitive regions as in MACHINESEG. In MACHINESEG, merging is controlled by edge strength only. In PQ, a model-based merit function is employed to determine which region should be merged next. This is more flexible than a function based on intrinsic image features without consideration of object types.

6.4. Selfridge

[Selfridge, 1982] describes a three-level system (model expert, segmentation expert, and parameter expert) to locate houses and roads in aerial photographs. The model expert represents objects and their expected locations. The segmentation expert selects a segmentation procedure for a given situation. The parameter expert searches for the best parameters of a thresholding algorithm to extract a region matching the expected appearance.

Performance evaluation of success and failure within each level is integrated into the analysis. If one rule fails, another is tried. The program terminates with failure only when all rules have been exhausted.

6.5. Hwang

SIGMA [Hwang, 1985] employs an approach similar to Selfridge, but emphasizes evidence accumulation. Blobs and ribbons which may be instances of houses and roads are initially extracted from the image using simple segmentation techniques. These instances are used to predict the locations of others. Computational resources are focused on those parts of the image that have high likehood of containing additional objects of interests. The likehood is based on the intersections of windows regions of interest associated with the hypotheses generated by the system.

Unlike SIGMA, PQ does not assume that there is a direct correspondence between the regions computed by segmentation and object model. PQ emphasizes the role of model-directed search for interpreting the results of image segmentation.

7. SUMMARY

Effective image segmentation is crucial to any image understanding system. Ideally, the segmentation should be sufficiently good to reliably determine the correspondence between parts of the images and parts of the object types of interest.

The approach of this investigation has been to initially segment an enhanced image into small, homogeneous regions, then to search among these for combinations that can be interpreted as object instances. Our system, called PQ, has been applied to find instances of roads, houses, and their shadows in aerial photographs of suburban neighborhoods.

The three subsystems of PQ are a high-level vision system (HLVS), representing goals and knowledge about objects, a low-level vision system (LLVS), for segmentation and evaluation of image predicates, and a graph search procedure to find instances by merging and testing.

The HLVS consists of descriptions of object types, a database of facts about the current image interpretation, and parameters of the initial segmentation and search for given object types.

The initial, uninterpreted segmentation of the enhanced image by the LLVS is represented by an attributed graph of connected regions, giving the adjacency relations and properties of these regions. This reduces the complexity of the image data, while preserving important details of contrast and geometry. A family of coarse-to-fine segmentations, obtained from the connected components algorithm by parameter adjustment, may be used in different searches.

The initial segmentation provides a basis for a search for object instances by merging components. This search procedure employs heuristics, including a merit function for expansion based on expected object grey value, as well as acceptance, optimization, and stopping criteria, derived from prior knowledge of object types.

The efficiency of the search procedure is improved by selecting a suitable initial segmentation, and by restricting search based on known spatial relations among objects.

It might be interesting to consider methods of evaluating evidence which having intermediate truth-values. A three-valued logic, with true, false, and indeterminate values, might be psychologically plausible, and useful for gradual interpretation of images. In additional, it might be useful to implement a more general search procedure employing window functions, recursive definitions, and finite quantification. The latter would make it possible to define searches for specific numbers of instances, for example, for exactly four corners of a house.

The present system correctly and efficiently identified

most instances of houses and roads in low-contrast aerial photographs.

REFERENCES

[Barrow, 1976] Barrow, H. G. and Tenenbaum, J. M., MSYS: a system for reasoning about scenes, Technical Note 121, Artificial Intelligence Center, Stanford Research Institute (1976).

[Feldman, 1974] Feldman, J. A. and Yakimovsky, Y., Decision theory and artificial intelligence: I. A semantics-based region analyzer, *Artificial Intelligence* **5**, 349-371 (1974).

[Harwood, 1984] Harwood, D., Subbarao, M., Hakalahti, H., and Davis, L. S., A new class of edge-preserving smoothing filters, TR-1397, Center for Automation Research, University of Maryland, College Park, MD (1984).

[Hwang, 1985] Hwang, V. S. S., Davis, L. S., and Matsuyama, T., Hypothesis integration in image understanding systems, TR-1513, Center for Automation Research, University of Maryland, College Park, MD (1985).

[Marr, 1982] Marr, D., Vision, W. H. Freeman Co. (1982).

[McKeown, 1984] McKeown, D. M. and Denlinger, J. L., Map-guided feature extraction from aerial imagery, *Workshop on Computer Vision: Representation and Control*, Annapolis, MD, 205-213 (1984).

[Nazif, 1984] Nazif, A. M., and Levine, M. D., Low level image segmentation: an expert system, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **6**, 555-577 (1984).

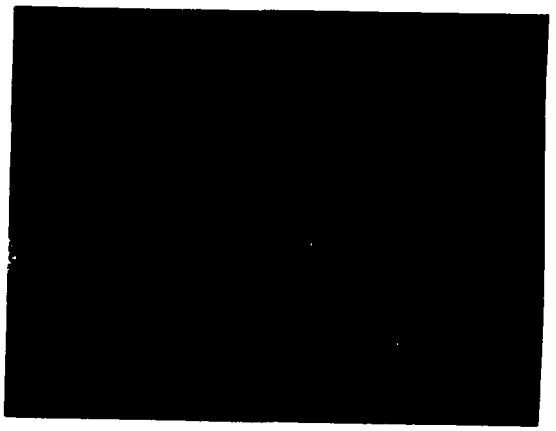
[Rich, 1983] Rich, E., Artificial Intelligence, McGraw-Hill (1983).

[Rosenfeld, 1982] Rosenfeld, A. and Kak, A. C., Digital Picture Precessing, Academic Press (1982).

[Rosenfeld, 1984] Rosenfeld, A., Image analysis: problems, progress and prospects, *Pattern Recognition* **17**, 3-12 (1984).

[Selfridge, 1982] Selfridge, P. G., Reasoning about success and failure in aerial image understanding, PhD Thesis, University of Rochester (1982).

[Tenenbaum, 1977] Tenenbaum, J. M. and Barrow, H. G., Experiments in interpretation-guided segmentation, *Artificial Intelligence* **8**, 241-274 (1977).

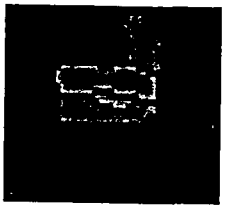


(a) Original image

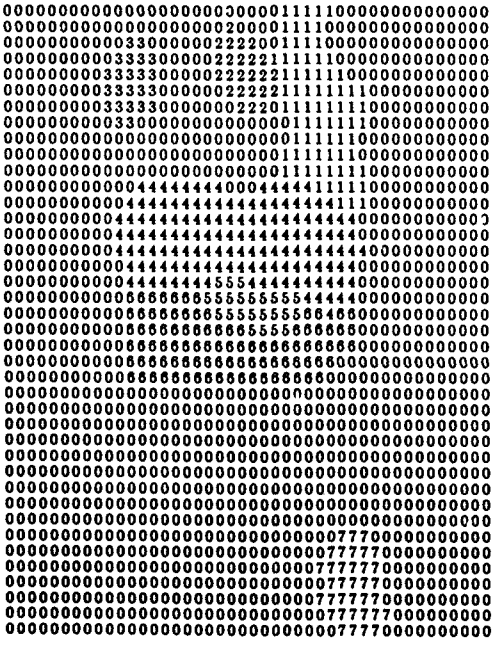
(b) Enhanced image

Figure 1: Original and enhanced image of two rows of houses

(a) Original image

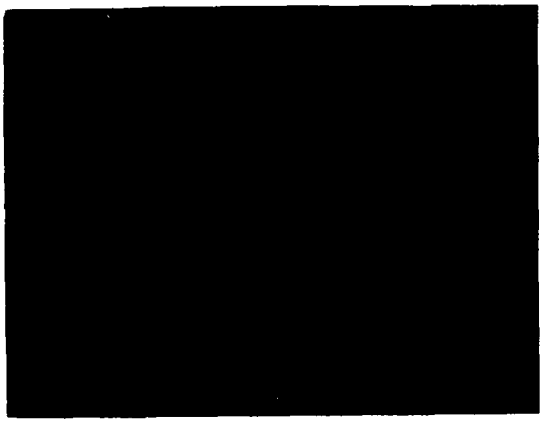


(b) Components



(c) Printout of component labels

Figure 2: Initial segmentation of a house and yard

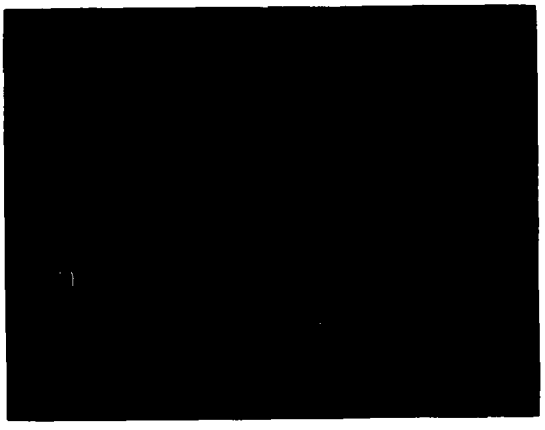


(a) Original image

(b) High-contrast segmentation

(c) Low-contrast segmentation

Figure 3: High- and low-contrast segmentations of neighborhood

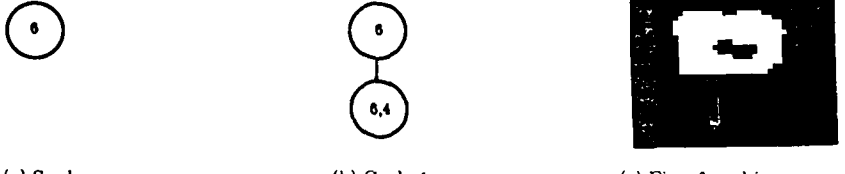


(a) Enhanced image

(b) Components

(c) Property and adjacency table

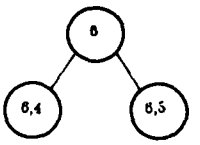
Figure 4: Initial segmentation and table of components: Properties and adjacencies for the house and yard image



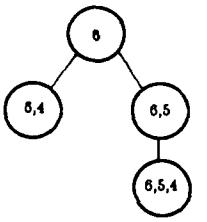
(a) Seed (b) Cycle 1 (c) First-found instance

Figure 5: FF search for first-found house instance

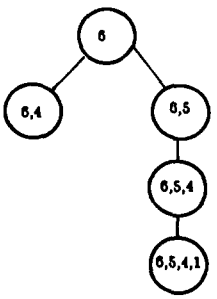
(a) Seed



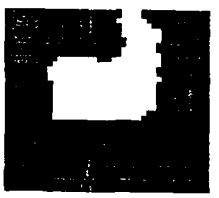
(b) Cycle 1



(c) Cycle 2

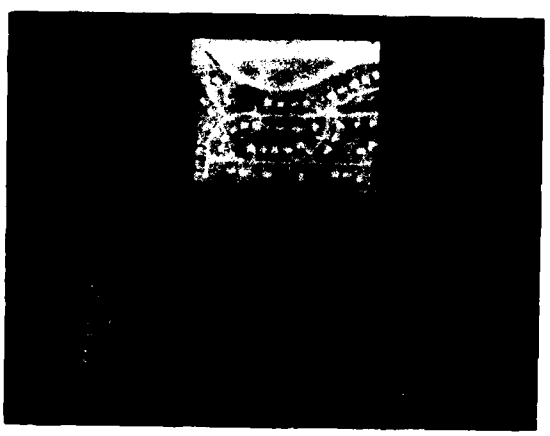


(d) Cycle 3



(e) First-large instance

Figure 6: FL search for first-large house instance

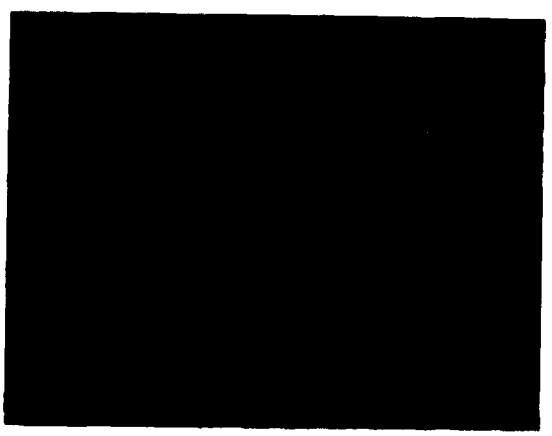


(a) Original image

(b) House instances without model adjustment

(c) House instances with model adjustment

Figure 7: Reinterpretation with adjusted model parameters



(a) Enhanced image

(b) First found house instance

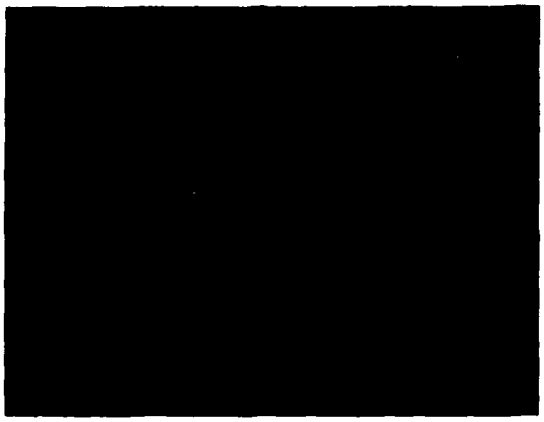
(c) House instance after optimization

(d) Original image

(e) First-found house instances

(f) House instances after optimization

Figure 8: Optimization of house instances



(a) Enhanced image

(b) House instances

(c) Shadows as supporting evidence

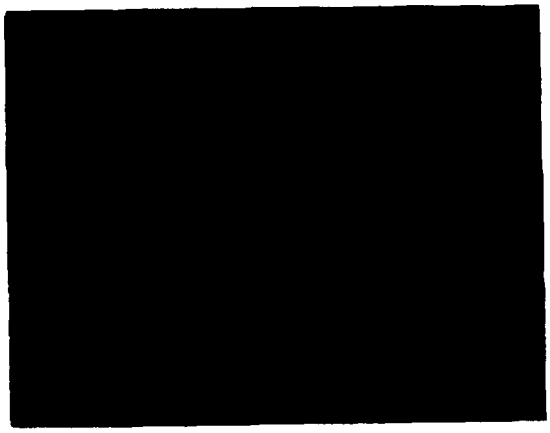
Figure 9: Search for supporting evidence

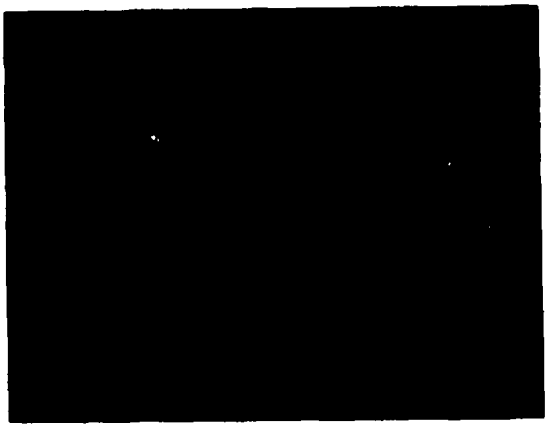
(a) Original image

(b) House instances

(c) Recovered parts of houses

Figure 10: Search for missing objects





(a) Original image

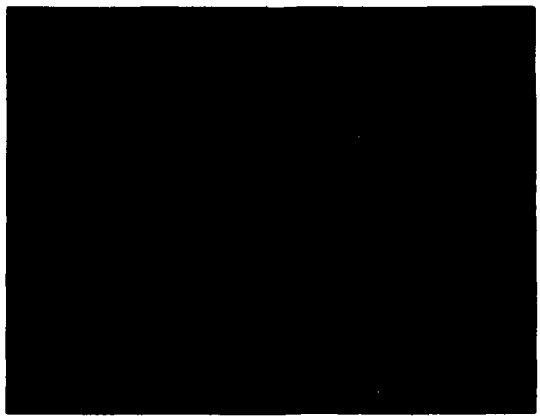
(b) Components for threshold 12

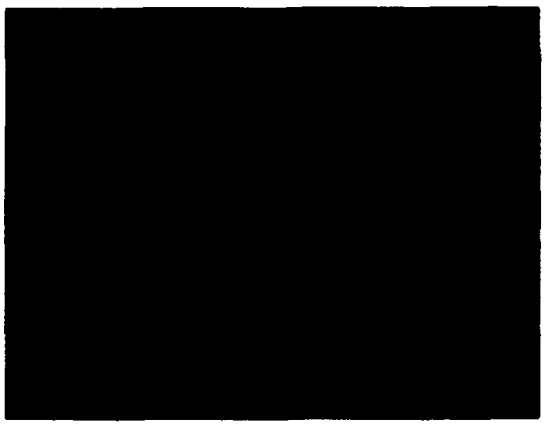
(c) House instances

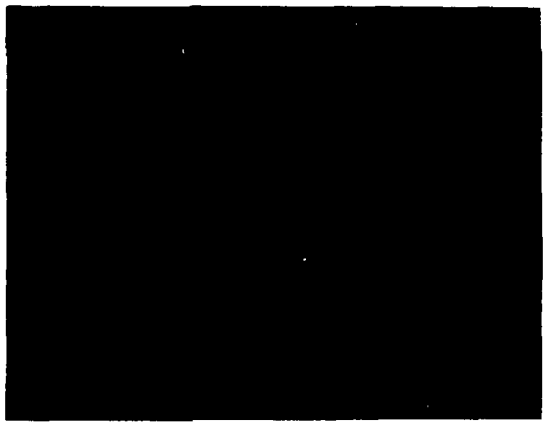
(d) Components for threshold 6

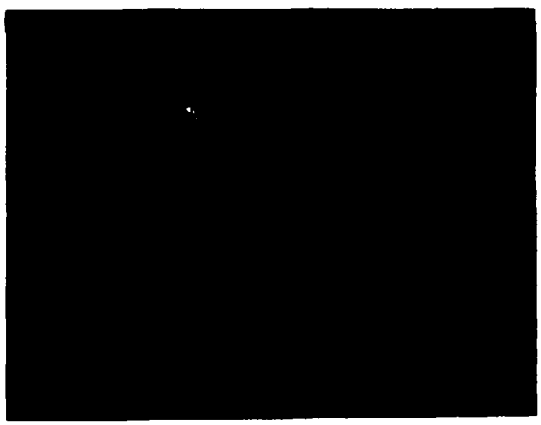
(e) Road complex instance

Figure 11: High- and low-contrast segmentations for houses and roads









(a) Original image

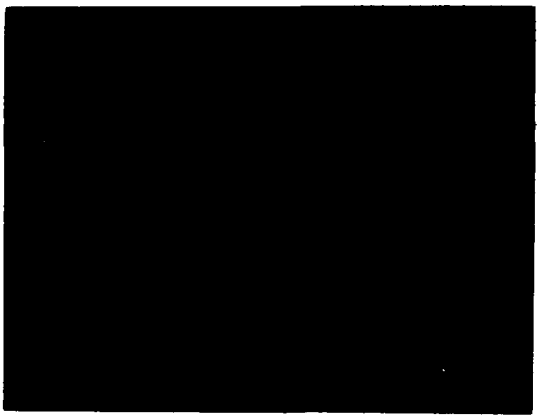
(b) High-contrast segmentation

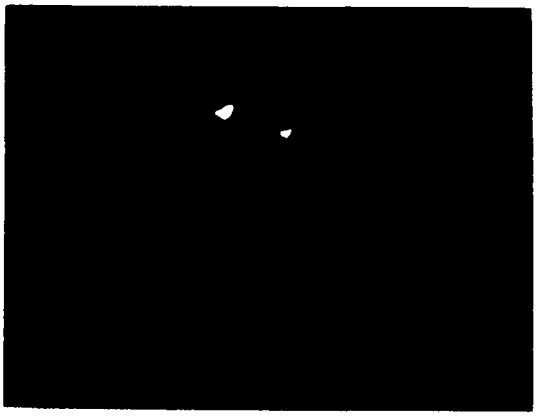
(c) House instances

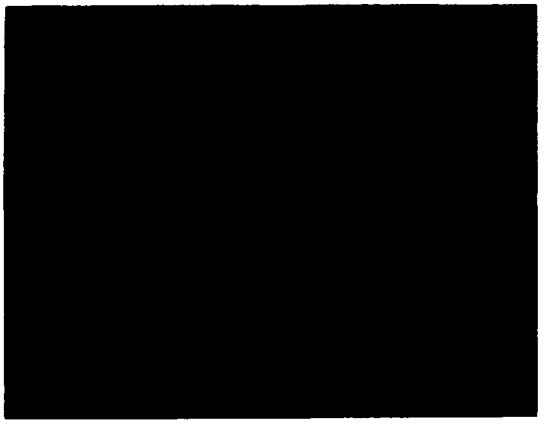
(d) Low-contrast segmentation

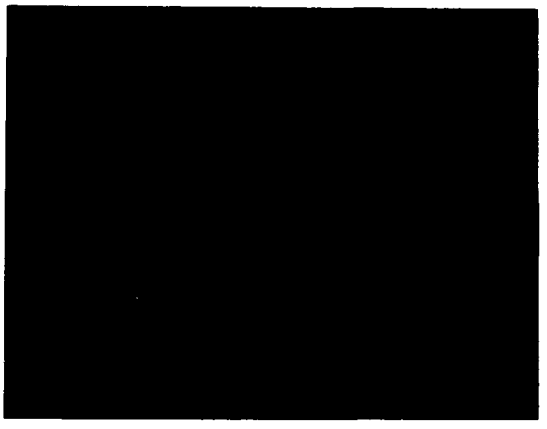
(e) Road instances

Figure 12: High- and low-contrast segmentations for houses and roads









(a) Original image

(b) Initial segmentation

(c) Unoptimized house instances

(d) Optimized house instances

Figure 13: FF house instances before and after optimization

(a) Original image

(b) Initial segmentation

(c) Unoptimized house instances

(d) Optimized house instances

Figure 14: FF house instances before and after optimization

Initial Hypothesis Formation in Image Understanding Using an Automatically Generated Knowledge Base *

Nancy Bonar Lehrer George Reynolds

Joey Griffith
University of Massachusetts at Amherst
Department of Computer and Information Science
Amherst, Massachusetts 01003

Abstract

This paper presents a method for initial object hypothesis formation in image understanding where the knowledge base is automatically constructed given a set of training instances. The hypotheses formed by this system are intended to provide an initial focus-of-attention set of objects for a knowledge-directed, opportunistic image understanding system whose intended goal is the interpretation of outdoor natural scenes. Presented is an automated method for defining world knowledge based on the frequency distributions of a set of training objects and feature measurements. This method takes into consideration the imprecision (inaccurate feature measurements) and incompleteness (possibly too few samples) of the statistical information available from the training set. A computationally efficient approach to the Dempster-Shafer theory of evidence is used for the representation and combination of evidence from disparate sources. We chose the Dempster-Shafer theory in order to take advantage of its rich representation of belief, disbelief, uncertainty and conflict. A brief intuitive discussion of the Dempster-Shafer theory of evidence is contained in Apendix A.

1. The Interpretation Problem

An important task in image understanding is to develop a mapping from low-level image events (such as regions, lines or surfaces) to higher level semantic abstractions (such as road, grass and foliage). Achieving this task requires developing a knowledge base which defines these semantic abstractions in terms of the low-level image events and using constructive techniques to match or correlate primitive features of the low level events against the knowledge base for each semantic abstraction. An inference technique is required to compare, contrast and combine match scores to create a consistent interpretation. Some schemes for combining information from various sources in initial hypothesis formation include additive "voting" methods [Hans86,Belk86], Dempster-Shafer pooling of evidence [Low82,Wes84,Rey86b], Bayesian methods [Duda73], constraint propagation [Kitc84,Wal72], as well as many ad hoc but heuristically adequate Mycin type systems [Shor76]. In any scheme three questions must be answered: What is What method will be employed for combining and propagating evidence?

In this paper we present a method for initial object hypothesis formation in image understanding that has evolved from earlier work in the VISIONS system environment documented in [Hans87]. These hypotheses can then be used as a focus-of-attention set by a knowledge-directed interpretation system and expanded into a more complete interpretation. An automated method is presented for defining a knowledge base based on the frequency distributions of a set of training objects and feature measurements. The system takes an image that is segmented into closed boundary regions and "matches" each region against the stored knowledge base to generate a set of initial hypothesis for a given region. This method can also be used to classify lines, surfaces or any other image abstraction or combination of image abstractions. In our formalism, evidence is represented by a *plausibility function* and combined using a computationally efficient approach to the theory of evidence as pioneered by Glen Shafer referred to as the Dempster-Shafer theory of evidence [Demp67,Shaf76].

At the heart of the Dempster-Shafer formalism is a rich representation of evidence, a belief function or mass function, and a method for combining evidence, Dempster's Rule. The formalism is often criticized for the computational cost associated with Dempster's Rule; in addition the Dempster-Shafer theory does not address the issue of acquiring mass functions. The system presented here addresses both these problems. It is able to use the rich semantics implied by the evidential representation of the Dempster-Shafer formalism without the computational cost associated with Dempster's Rule as well as defining an automatic method for generating a knowledge base. A brief intuitive discussion of the fundamental principles of the Dempster-Shafer formalism are presented in appendix A.

2. The VISIONS Experiments

In general we can think of intermediate-level image abstractions as *tokens* one or two steps removed from the raw image data represented as pixels. Regions can be thought of as area filling abstractions connecting pixels with some

*This work has been supported by the following grants: Air Force Office of Scientific Research 86-0021, the Defence Mapping Agency 800-85-C-0012, and the National Science Foundation DCR-8318776.

homogeneity constraint [Kohl84]. Straight lines may connect and group pixels with the same gradient direction [Burn86,Weis86]. Each intermediate level token can be associated with a feature vector that measures primitive features. If pixels are grouped into closed area-filling regions, measures can be defined which statistically describe a region's color, intensity or texture. Depending upon the line algorithm used, lines also have a variety of primitive features such as length, position in the image (rho), angle or orientation (theta), and measures of the contrast relating the values of pixels on either side of the line.

An approach taken in VISIONS [Hans87] is to use these primitive features to create initial hypothesis labels which associates with each region a semantic label to bootstrap the high-level interpretation process. Although regions are being labeled in this approach, line data can be incorporated into the interpretation process by the use of relationships between lines and regions [Belk86].

2.1 Approaches to Representing and Combining Evidence

Existing approaches to the generation of initial hypotheses have used interactive and heuristic approaches to the knowledge engineering problem. One in particular, the rule-based object hypothesis system of Hanson and Riseman [Hans86], defines constraints on the features of line and region image abstractions; it uses frequency distributions to guide the formation of heuristically defined, piecewise linear ranking functions called *rules*. In that system, rules provide a vote for a specific object, and a set of "simple" rules are combined via a weighted average to form a "complex" rule which can then be similarly combined. The output of a complex rule represents a weighted combination of evidence from various features and is used to rank order image regions (or collections of regions and lines) according to how well they match a prototype object.

The contribution of the system presented here is two fold: (a) a formal and theoretical foundation for the combination and representation of evidence, and (b) the automatic construction of the knowledge base. It defines the knowledge base automatically using statistical information obtained from a set of training object instances and uses a computationally efficient approach to the Dempster-Shafer theory of evidence for the representation and combination of evidence from disparate sources. The knowledge base is represented in terms of *plausibility distributions*; their construction takes into consideration the imprecision (i.e. inaccurate feature measurements) and incompleteness (i.e. too few samples) of the statistical information available from the training set. We show that the automatically generated plausibility distributions characterize the range of feature values associated with each object in the training set without biasing the interpretation towards objects more likely to appear at random in the training set. The term plausibility is used to indicate the equivalence between our formalism and the semantics and functionality of the term *plausibility* in the Dempster-Shafer formalism.

2.1.1 The Rule System

The task of the rule-based object hypothesis system of Hanson and Riseman [Hans86] system is to provide a set of candidate object hypotheses to a knowledge-directed schema network. Based on these initial hypotheses an island driving focus-of-attention strategy is employed to initiate further processing. Rules are formed by heuristically assigning a vote for a specific object to ranges of feature values. Sets of "simple" rules are combined via a weighted average from "complex" rules. Sets of complex rules are then combined in the same way to form the initial hypotheses. A rule is represented as six threshold values, $\theta_1, \theta_2, \ldots, \theta_6$, these thresholds define a piecewise linear mapping function from feature space to object space. The intervals $[-\infty, \theta_1]$ and $[\theta_6, \infty]$ represent a veto range, $[\theta_3, \theta_4]$ the range where the object label associated with the prototype feature vector receives a maximum vote of 1, the intervals $[\theta_1, \theta_2]$ and $[\theta_4, \theta_5]$ a noncommittal vote of zero, and finally $[\theta_2, \theta_3]$ is linearly ramped from 0 to 1 whereas $[\theta_5, \theta_6]$ is ramped from 1 to 0. Figure 2.1 shows the construction of a simple rule. The weights given to the combination rule are heuristically defined on a scale from 1 to 5.

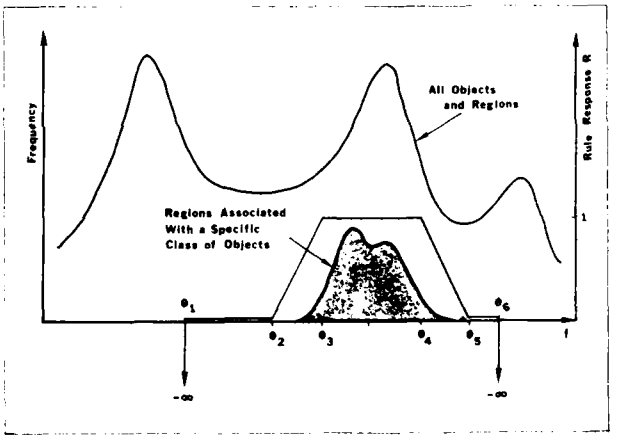


Figure 2.1: Structure of a Simple Rule
Structure of a simple rule for mapping an image feature measurement f into a score for a label hypothesis on the basis of a prototype feature value. The object specific mapping is parameterized by seven values, $f_p, \theta_1, \ldots, \theta_6$.

Due to the potentially large number of rules needed for each object in the initial hypothesis set, the rule system is designed to simplify the definition of each rule. The user choosing the threshold values θ_1 through θ_6 for a particular object is equipped with an interactive environment for constructing these rules and displaying their effect. In this system each object has a different set of features which contribute to the objects prototype feature vector. Given an image token, the score for one object is never directly compared against the score for another object; instead for

each object, tokens are ranked by how well they score for that object. Regions with the highest scores for a particular object are used as exemplar regions in an island driving strategy applied in later stages of the interpretation.

The rule system was developed in reaction to the problems of using Bayesian techniques for the classification of tokens based strictly on statistical information available from an inadequate training set. In particular, if the training set is small, the feature distribution may only coarsely characterize feature space; in addition the a priori probability of seeing a particular object at random in the training set is also a highly unreliable estimate of seeing that object in the world, yet plays a powerful role in bayesian decision functions [Duda73,Wood78]. Section 4.1.1 discusses in more detail the problems inherent in Bayesian methods when the statistical samples contain inaccurate or imprecise information, see also [Low82,Wes84].

2.1.2 The Plausibility Formalism

The approach used in our plausibility formalism differs in many aspects from the rule system. Whereas the high level goal of producing object hypotheses for a knowledge directed schema network is common to both systems, our approach is not thought of as producing a ranking of regions for each symbolic object (although the results may ultimately be used in this manner). The outcome is instead viewed as associating with each region an evidential model of the current state of interpretation.

Our general approach resembles the principles behind a least commitment or constraint propagation system. The system uses a set of features to rule out possible object hypotheses until only a small set of initial hypotheses remain. Each piece of evidence results in an assignment of a *plausibilty value* to each hypothesis, yeilding a *plausibility function* defined on the set of hypotheses. Each plausibility value is generated by comparing a feature value for an image event against the knowledge base. For example, the system may measure the average color of region 2; the value returned is then compared with the knowledge base for each of road, foliage, sky... to find a plausibility value for each possible semantic abstraction. The plausibility value represents the amount to which that hypothesis should not be ruled out as a possibile hypothesis. Each plausibility function represents a mass function (a method for transforming a plausibility function into a mass function is discussed in Section 3.). The plausibility functions derived from each feature measurement are combined such that they represent the plausibilities of each singleton hypothesis as if the equivelent mass functions were combined using Dempster's Rule. The result is a combined plausibility function which represents a consensus of opinions from disparate pieces of evidence. Figure 2.2 shows the construction of a plausibility function given a feature measure and a knowledge base.

It is important to understand that the plausibility functions returned by each knowledge source are not statements about the *probability* of a hypothesis in a Bayesian view. The role of a plausibility value is to rule out unlikely states of nature. The minimum semantic requirement for a plausibility value is that it represent the extent to which a state of nature should not be ruled out given an event.

2.1.3 Differences in Evidential Representation

In the rule system, a vote can have one of three different effects on an object hypothesis. If the vote is between zero and one, this supplies supporting evidence, a vote of $-\infty$ vetoes an object hypothesis altogether, and a vote of zero is noncommittal. The final score represents supporting evidence for a particular object. On the surface, the plausibility formalism only allows ruling out hypotheses, but by transforming a plausibility function into a mass function after combination we are also able to represent supporting evidence as well as the possiblity that an image abstraction may not be any of the objects represented by the set of possible hypotheses, i.e. conflicting evidence.

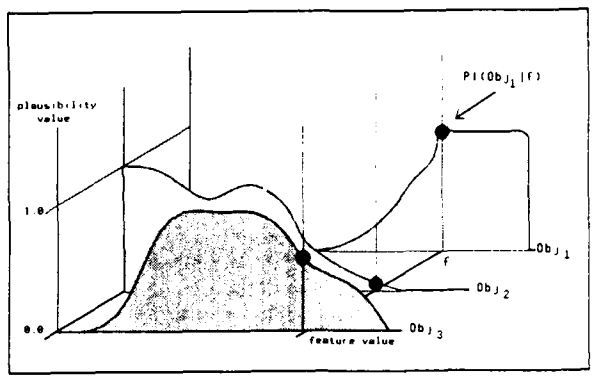


Figure 2.2: Constructing Plausibility Functions
The three shaded curves represent the knowledge base for Obj_1, Obj_2, and Obj_3 in the form of *plausibility distributions* for some feature. Given a feature value f, the plausibility value $Pl(Obj \mid f)$ can be determined from the plausibilty distribution as shown by the shaded circles. A plausibility function is the set of plausibility values obtained for a given feature value.

The greatest advantage of the Dempster-Shafer approach is its representation of several different types of evidence: supporting evidence, plausible evidence, and conflicting evidence. It greatest disadvantage is the exponential nature of Dempster's Rule and the explicit representation of the powerset of all object hypothesis. Our system provides a technique that allows the representative power of the Dempster-Shafer approach to be combined with the computational efficiency of the rule system.

3. A Computationally Efficient Approach to Dempster-Shafer

A major criticism of the Dempster-Shafer formalism has been the combinatorial problems related to use of the powerset of all possible hypotheses in Dempster's Rule. As the frame of discernment becomes large, the computation

becomes unmanageable. To overcome this combinatorial problem, needed is a way to represent a mass function using only the elements of the frame of discernment and a combination rule that is equivalent to Dempster's Rule for this simplified representation. Reynolds et al. [Rey86b] describe a method whereby knowledge sources need not return a mass function, but rather a *plausibility function* where each individual element of the frame of discernment, Θ, is assigned a *plausibility value* between zero and one. A mass function representation for a given plausibility function can be obtained using the formula:

$$m(A) = \frac{\prod_{a \in A} pl(a \mid f) \prod_{a \in \neg A} (1 - pl(a \mid f))}{1 - k}. \quad (3.1)$$

where $pl(a \mid f)$ is the plausibility of seeing object a given a feature value f (see figure 2.2). The conflict value of a plausibility function is defined as

$$k = \prod_{a \in A} (1 - pl(a \mid f)). \quad (3.2)$$

Given two plausiblity functions $pl_1(a \mid f_1)$ and $pl_2(a \mid f_2), a \in \Theta$, we define the combination $pl_3(a \mid f_1 \wedge f_2)$ by the rule

$$pl_3(a \mid f_1 \wedge f_2) = \frac{pl_1(a \mid f_1) \cdot pl_2(a \mid f_2)}{1 - k}, a \in \Theta, \quad (3.3)$$

where k is the conflict value defined above for $pl_3(a \mid f_1 \wedge f_2)$ Analogously we define the combination of an arbitrary number of plausiblity functions.

It is shown in [Rey86b] that this combination rule produces the the plausibilities of the singletons, as defined in the Dempster-Shafer theory, when the mass functions generated by formula 3.1 are combined using Dempster's Rule. The terms *plausibility* values and *plausibility* functions were chosen to point out this equivalence. A complete discussion and proof of the equivalence of this simple representation and multiplicative rule with a class of mass functions and Dempster's Rule can be found in Reynolds et al. [Rey86b].

4. Defining the Knowledge Base

In any image understanding scheme one of the most ill-defined tasks is that of defining the knowledge base. To overcome the problems of an inaccurate training set, the rule system creates its knowledge base by hand using the histogram of a set of training instances for heuristic guidance. On the other hand, Bayesian techniques address the problem of efficient definition of object rules for a given feature in terms of their conditional probability distribution, but these techniques lack the ability to handle inacurate or insufficient sample sets. In this section is to discussed a method for building a knowledge base, called *plausibility distributions*, determined directly by the statistics of a training set of image primitives (in this sense similar to bayesian techniques), but which can also deal with certain kinds of uncertainty inherent in the training set. In particular we address the situation where the training set is sufficient to show where an object lies in feature space, but the a priori estimate of seeing that object in the world, as estimated by the number of samples in the training set, is inadequate with regard to identifying objects in feature space.

4.1 Criteria for Plausibility Distributions

The approach taken by Hanson and Riseman [Hans86] describes a method for heuristically assigning a rule score based on the feature distributions for a particular object in relation the feature distribution for the entire image (or set of images). Our intent is to use statistical information to *automatically* constructs a knowledge base of plausibility distributions given information information about the frequency distribution of a set of training instances.

The need is to find some function which characterizes the area in feature space in which an object lies and also takes into account the relationship between the object and total world sample. If we pick a set of pixels to represent some object, and a set of features over which to characterize each object, then we have the following statistical information: the frequency of seeing a particular feature value, $h(f)$ and the frequency of seeing a particular feature value and a particular object, $h(f \wedge o)$. (See Figure 4.1.)

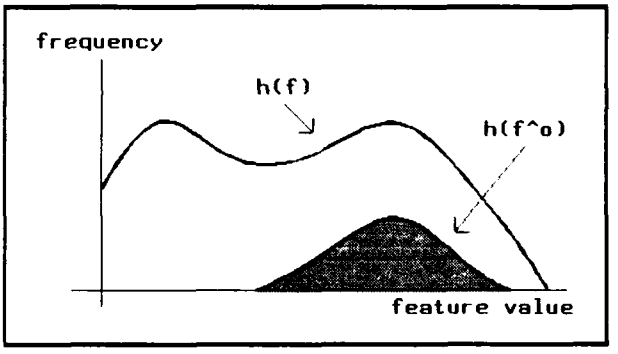


Figure 4.1: Frequency distributions
The larger curve represents the frequency of seeing a particular feature value, denoted by $h(f)$. This is the distribution of the feature over the entire population of samples. The smaller, shaded, curve represents the frequency of seeing a particular feature value and a particular object, $h(f \wedge o)$, the distribution of the feature over only those samples labeled object o.

Some considerations in evolving a system for producing automatic plausibility distributions include:

Statistical Accuracy: What statistical information is available from a training set of object instances, what assumptions can be made about the accuracy of these statistics, and how useful is this information for discrimination and recognition?

Semantics of a Plausibility Value: What are the specific semantics of a *plausibility value* with respect to our plausibility formalism.

Size of the Training Set: Should the final plausibility distribution be independent of the number of positive training instances for a particular object in the world? For example, if 30% of the training instances are $object_i$ and 20% are $object_j$, should the relative *plausibility* of finding $object_i$ over $object_j$ be dependent upon the probability of picking $object_i$ out of the training set at random (this is the classic discussion about the validity of a priori statistics)? It may or may not be reasonable to assume a priori that certain objects are more likely to be seen than others.

We briefly discuss each of these considerations in the next three sections.

4.1.1 Statistical Accuracy

We must first address the question of the accuracy of statistics collected over a set of training instances. Any matching or inference technique is only as good as the representation of the world knowledge as defined by a training set, and the quality of the information returned by the processes which measure the features to be compared with the world knowledge. In complex domains, both areas are subject to uncertain, imprecise and occasionally inaccurate information [Low82]. One concern with probabilistic methods is the amount of information needed to assure quality statistics. To accurately compute the conditional probability of seeing a particular state of nature ω given a feature measure, $p(\omega \mid f)$, requires the a priori knowledge of seeing ω given no other information $p(\omega)$, the probability of perceiving a particular feature measure $p(f)$, and the probability of seeing a particular feature measure given a state of nature $p(f \mid \omega)$. At best, this information is difficult to obtain or reliably estimate. Sources of uncertainty can arise through innacurate feature measures, incomplete data sets, aliasing resulting from digitization and region segmentation, and inacurate region segmentation.

4.1.2 Semantics of a Plausibility Value

Secondly, we look at the desired semantics of a plausibilty value. The objective at this stage of interpretation cannot be emphasized enough. When the interpretation begins, each token is possibly any object contained in the frame of discernment or the unknown event, moreover the features used to characterize object hypothesis at this level are extremely primitive. Therefore the goal is not object classification, but rather a pruning of possible hypotheses. With this in mind, the minimum semantic requirement of a plausibility value is that it represent how much an object should not be ruled out given a particular feature measure.

4.1.3 Size of the Training Set

Finally, some understanding must be reached about the size of the sample set for each object in the frame of discernment. Using Bayesian techniques, the conditional probability $p(\omega \mid f)$ represents both the probability of seeing object ω, $p(\omega)$, and knowledge about the probability of seeing feature measure f given ω, $p(f \mid \omega)$.

$$p(\omega \mid f) = \frac{p(f \mid \omega)p(\omega)}{p(f)}.$$

Consider the two object case, with objects ω_1 and ω_2. Given equal probability for $p(f \mid \omega_1)$ and $p(f \mid \omega_2)$ then the decision process is based solely on the the values for $p(\omega_1)$ and $p(\omega_2)$. In our paradigm of using sample regions hand labeled as the training set, then these two a priori probabilities are based solely on the number of samples for any object in the training set. This is not an adequate estimate. A compromise has been suggested that the a priori probability $p(\omega)$ is assumed equal for all objects and thus can be removed from the calculation leaving the likelihood ratio,

$$\frac{p(f \mid \omega)}{p(f)}.$$

A ratio greater than 1 indicates that the feature measure f is more likely to occur for this particular object than for some random object. A ratio equal to 1 indicates no information, and less than 1 states that the feature value f is more likely to occur by chance for any object than it is for a particular object ω_1. This ratio is the theoretical foundation underlying the heuristic approach of Hanson and Riseman [Hans86] and has also been employed by Woods in speech understanding [Wood78]. If the a priori statistics are assumed to be innacurate, this is still not a reasonable solution because the a priori statistic $p(\omega)$ is still present in the term $p(f \mid \omega)$.

If it is reasonable to assume that certain object are more likely to appear than other object, then some estimate of the a priori probability may be devised. On the other hand, if you assume that the a priori probabilities are ill defined at best, then it is not good enough to simply remove that term from the computation of the conditional statistic, rather some effort must be made to factor out the size of the sample over which the probabilities are calculated.

In the domain considered here the number of samples for a given object, o, is not be an adequate estimate of the a priori probability of seeing that object in the world. Indeed this a priori probability may be impossible to determine. However the number of samples does influence $h(f)$ and $h(f \wedge o)$. What is needed is to find the range of feature values associated with a given object and to factor out any effects related to the probability of picking an object at random out of the training set.

In summary, early work on the generation of feature rules by Hanson and Riseman suggests that the rule for an object should be influenced not only by the its conditional

feature distribution, but also by its relationship to the feature distribution of all the objects in the entire training set. In addition, we set as our goal that the final plausibility distribution should be characterizations of feature ranges and not include information about a priori probabilities.

4.2 Automatic Generation of Plausibility Distributions

For a given feature value consider the ratio of the height of a frequency distribution for a given object, $h(f \wedge o_i)$, to the height of the frequency distribution for all training objects, $h(f)$ (see Figure 4.1). This is by definition an estimate of the conditional probability [1]

$$\hat{p}(o_i \mid f) = \frac{h(f \wedge o_i)}{h(f)}. \tag{4.1}$$

Similarly defined are the estimates $\hat{p}(o_i)$, the relative frequency of seeing $object_i$ in the knowledge base and $\hat{p}(f)$, the relative frequency of seeing a particular feature value f.

$$\hat{p}(o_i) = \frac{\#of object_i}{\sum_{j=1}^{n} \#of object_j} = \frac{\sum_f h(f \wedge o_i)}{\sum_f h(f)} \tag{4.2}$$

$$\hat{p}(f) = \frac{h(f)}{\sum_f h(f)} \tag{4.3}$$

As mentioned earlier, one desirable characteristic of a plausibility distribution is that it be relatively invariant with respect to the size of the training set used to model the feature distribution. That is, two distributions differing only in the size of the sample set over which they are defined should have the same plausibility distribution. We can show that one way of accomplishing this behavior is to use the estimate of the a priori probability, $\hat{p}(o_i)$, as a decision threshold on the conditional probability $\hat{p}(o_i \mid f)$. If the value of $\hat{p}(o_i \mid f)$ is at least as large as the estimate of seeing o_i, $\hat{p}(o_i)$, assume a plausibility value of 1, otherwise normalize by $\hat{p}(o_i)$ We now have the following definition for a plausibility function.

Definition: *For each object o_i and each feature f we define a plausibilty value for object o_i given feature f as follows:* [2]

$$Pl(o_i \mid f) = Min(1.0, \frac{\hat{p}(f \wedge o_i)}{\hat{p}(f)\hat{p}(o_i)}). \tag{4.4}$$

We can now examine the behavior of a plausibility value with respect to the size of the training set that makes up our statistical samples. To do this we must look at the behavior of

$$\frac{\hat{p}(f \wedge o_i)}{\hat{p}(f)\hat{p}(o_i)} \tag{4.5}$$

with respect to two objects with the same distribution but with difference sample sizes. Given two objects, o_1 and o_2, we can show that $Pl(o_1 \mid f) = Pl(o_2 \mid f)$ regardless of the difference in the size of $\hat{p}(o_1)$ and $\hat{p}(o_2)$ as long as the distibution for the two objects occupy the same area in feature space.

Theorem: *Let $\hat{p}(f \wedge o_2) = \alpha\hat{p}(f \wedge o_1)$ and $\hat{p}(o_2) = \alpha\hat{p}(o_1)$ where α is some scalar, then*

$$\frac{\hat{p}(f \wedge o_2)}{\hat{p}(f)\hat{p}(o_2)} = \frac{\alpha\hat{p}(f \wedge o_1)}{\hat{p}(f)\alpha\hat{p}(o_1)} = \frac{\hat{p}(f \wedge o_1)}{\hat{p}(f)\hat{p}(o_1)} \tag{4.6}$$

We show in figure (4.2) that this function indeed characterizes the proportion of feature space in which the objects feature distribution lies and minimizes the effect of the sample size over which the statistics are taken as stated by the above theorm.

5. Using Plausibility Functions for Initial Hypothesis

In this experiment we started with six images of New England road scenes digitized to a resolution of 256 x 256 pixels. Each image was then segmented using a knowledge based segmentation technique [Hans87]. Each region was hand labeled as one of the following labels {ambiguous, barn, dirt, foliage, grass, gravel, house, phonepole, pole, post, railing, road, roadline, sign, sky, sky-tree, tree-trunk, wire}, but only objects with a significant number of occurrences in the training set were used in the frame of discernment. The context for this experiment is defined by the following question, frame of discernment and set of feature spaces:

Question: "What are the plausible semantic labels for this region?"

Frame Of Discernment: {foliage, grass, gravel, road, roadline, sky, sky-tree, trunk}. [3]

Feature Spaces: Four feature categories were used; Intensity, Color, Location and Texture.

- **Intensity Features:** Y color transform, Intensity color transform, Raw red, Raw green, Raw blue.
- **Color Features:** Percentage of red, Percentage of green, Percentage of blue, Excess red, Excess green, Excess blue, Hue, Saturation, Q color transform, I color transform.
- **Texture Features:** Horizontal edge measure, Vertical edge measure, Lower diagonal edge measure, Upper diagonal edge measure.
- **Location Features:** Row position of centroid.

[1] We will use the notation $\hat{p}$ to indicate the use of estimates of probabilities based on discrete samples.

[2] A full discussion of this definition and proof of related theorms can be found in [Rey86b].

[3] The unknown object is not explicit but rather implicitly represented by the conflict value k.

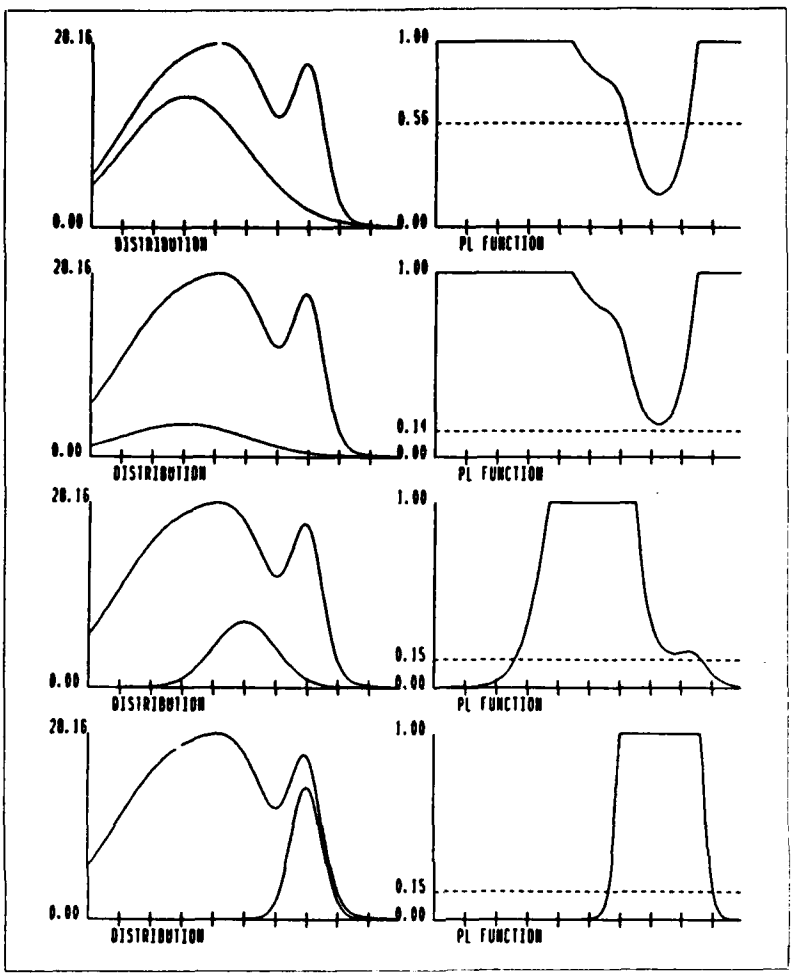


Figure 4.2: Plausibility distributions of synthetic world knowledge.
This graph shows the plausibility distributions for synthetic world knowledge as defined by a set of four gaussian distributions. On the left is displayed $h(f)$, the sum of the gaussians, with $h(f \wedge o)$ shown separately from top to bottom. On the right, the four plausibility distributions are displayed. Note that the top two distributions have the same mean and standard deviation, and their plausibility distributions are identical. The dotted horizontal lines pass through $\hat{p}(o)$.

No information about object size or shape were used in this set of experiments. The objective was to use a simple set of feature measures to reduce the set of possible object hypotheses for any particular region. In particular, the feature measures used here contain little or no special knowledge which relates to a specific context. The approach is not strictly classification, but rather to provide some initial evidence for an hypothesis. The initial hypothesis can then provide information about a more specific context to be used as interpretation continues; with this more specific context are more specific, perhaps more expensive, feature measures. Under current development is a system for extending and verifying an initial hypothesis using a high-level knowledge based system implemented in a black board architecture [Weym86,Drap86].

The knowledge base for each feature space was formed over the feature distribution of the hand labeled regions from all six images. The feature spaces were specifically designed to use only features that could be measured over pixels. For each feature, a feature plane is defined which encodes a feature value at every point in the image. The feature planes and the training regions are then used to create a pixelwise frequency distribution for each object in the frame of discernment. An alternative is to define the frequency distributions by the mean feature value defined over the training regions and to weight the mean value by the size of the region. In the latter method, a high degree of smoothing is required to produce a robust frequency distribution. In the pixelwise method only minimal smoothing was used. During the interpretation phase, each knowledge

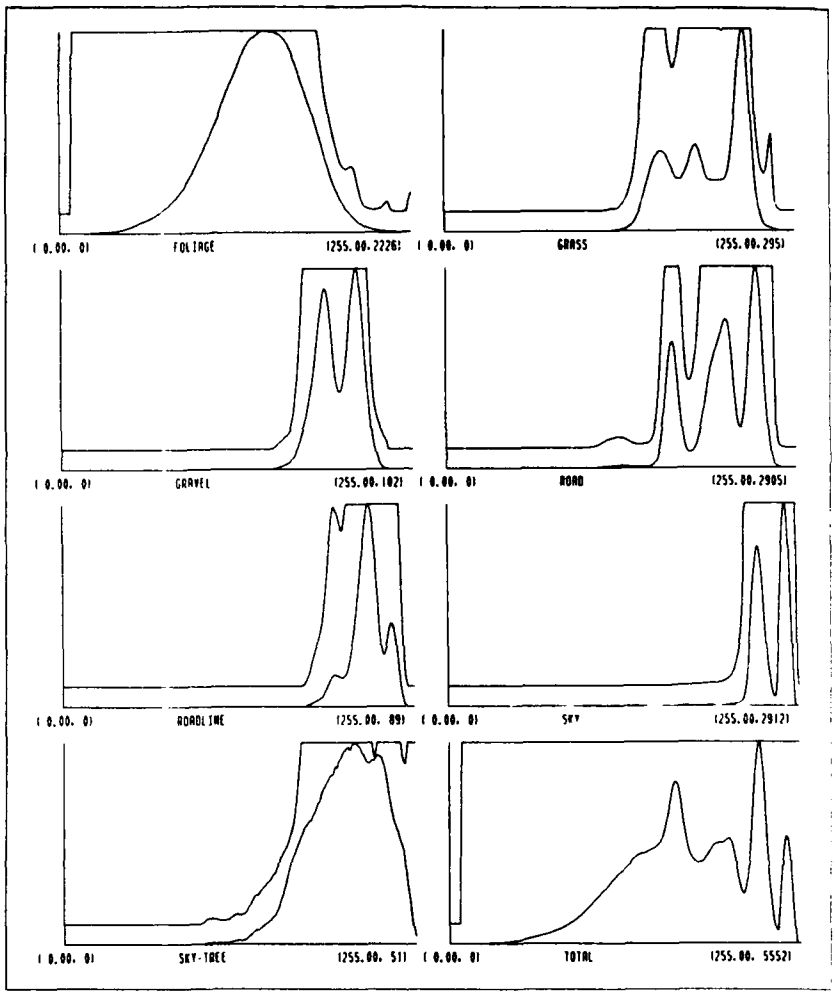


Figure 5.1: Plausibility distributions of Intensity.
These graphs show the feature histgrams and resulting plausibility functions for the objects foliage, grass, gravel, road, roadline, sky, sky-tree, and total. For each object, the lower curve is the histogram of the actual feature values for the regions in the training set. The upper curve is the resulting plausibility function. Each object histogram is scaled with respect to the number of samples contained in the training set for that particular object. The histogram for "total" in the lower right hand corner represents all regions in the training set including those whose labels are not included in the set of possible initial hypotheses.

source returns an evidential model based on the mean feature value of the region in question.

In addition, no plausibility was allowed to recieve a value of zero. Due to the multiplicative nature of our combining function, a value of zero could cause a hypothesis to be completely ruled out based on errorful information. Instead, all plausibility values were normalized between .1 and 1. Figure 5.1 shows the plausibility functions generated for the feature intensity, where intensity is defined as $R + G + B/3$.

5.1 Understanding Conflict

In Section 3. we discuss a combination function for plausibility functions which parallels the Dempster's Rule applied to mass functions. This combination function uses $(1-k)$ as a normalization constant. This section introduces other possible uses of normalisation and different nomalisation constants when conflict is due to a situation other than the disagreement of knowledge sources.

An approach taken in the rule system is to load the system with redundant information so that no one knowledge source contributes a significant amount of information to the interpretation process. This is desirable if any of the

knowledge sources are suspected to be unpredictably errorful. Unfortunately, in the approach presented here, as the number of objects and feature spaces increase, so does the conflict between plausibility functions in the interpretation of a token. In particular the automatic plausibility distributions may allow many objects to receive a small plausibility value, ruling out completely no one object for any one feature value. For many tokens, each object in the frame of discernment will be ruled out to some significant degree by at least one piece of evidence due to errorful data (e.g. innacurate feature measurement due to aliasing or poor placement of a segmentation boundry). This situation introduces conflict into the final interpretation which is not due to the detection of an unknown image event.

Using the normalization of $(1 - k)$ we are able to explicitly represent an unknown event by the value of k in each plausibility function. Intuitively, the amount to which two knowledge sources do not agree, k, indicates the amount to which the correct initial hypothesis is not contained in the frame of discrenment. Normalizing a combined plausibility function by $(1 - k)$ produces plausibility values identical to the plausibilities over the singletons of the equivalent mass function obtained using Dempster's Rule and allows an unknown event to occur and be represented as conflict.

It has been argued that normalization is used to eliminate or *hide* a contradiction of aggregate evidence [Zad83]. It must be noted that with a multiplicative combination function, the use of no normalization will create monotonically decreasing plausibility values. This means that any evidence supplying a plausibility value less than 1.0 can only detract from a hypothesis, and the more evidence that is accumulated, the more the system will become susceptible to errorful data. The use of a normalization by $(1 - k)$ however keeps the system from being a monotonically decreasing system, preserves the semantics of the plausibility values as the plausibilities of the singleton subsets of the related mass function, and preserves the representation of the unknown object as conflict.

5.2 Interpretation Strategies and Future Work

In these experiments, the Dempster-Shafer formalism is not used as an end to reasoning, but rather as a representation formalism for evidence. The interpretation strategy to be employed after initial hypothesis generation dictates some of the desired qualities for a decision over the set of initial hypothesis. One approach might be to use the initial hypothesis to indicate exemplar regions, regions which most look like grass for example. These regions can then be used to constrain the interpretation of neighboring regions, on the other hand, the initial hypothesis can be viewed as simply reducing the set of possible hypothesis.

Several different decision strategies were suggested creating the initial set of hyotheses for this experiment. The simplest strategy is to normalize out all conflict contained in a plausibility function (by dividing the plausibility function by the maximum plausibility value) and use these numbers for ranking objects. This strategy does not allow the representation of the unknown object. Normalizing by $(1 - k)$ allowed an explicit representation of the unknown event. Another strategy suggested uses the idea of consistent evidence. A simple strategy can be used to find the most likely object. Next, all plausibility functions which rule out to some significant degree the top ranking object, as produced by the simple strategy, are then removed from the final combination of evidence.

The decisive factor for determining the best decision strategy at any step in the interpretation should be the context and objectives of the interpretation system as a whole. There may be only a few relevant control (control about conflict) strategies, or control may become a highly variant context dependent process.

6. Results and Figures

The hypotheses displayed in this section are initial hypotheses to be used by a knowledge-directed, opportunistic image interpretation system. The initial hypotheses can verified and used to create a contextual environment in which to hypothesise objects not defined by the frame of discernment, hallucinate the merging or splitting of image tokens, and in general aid in the development of a mapping between image events and world events.

We chose the following two step combination scheme using the multiplicative combination function defined by formula 3.3:

1. Combine all the plausibility functions for each of the four feature spaces (i.e. intensity, color, texture and location) and normalize each by $(1-k)$, with k defined by for each plausibility function.

2. Combine the four resulting plausibility functions and again normalize by $(1 - k)$, with k defined by the combined plausibility function.

By combining the evidence for each category of features first, we were able let each group have equal weight in the final combination. For instance, there are ten color features, but only one location feature. By combining the color features and renormalizing before combining with the location feature, we were able to represent color and location equally in the final interpretation. Once the evidence contained in each plausibility function is combined, a mass function is constructed as discussed in Section 3. to represent the final evidential model.

Figure 6.2 through figure 6.6 each show four pieces of information for a particular object hypothesis. The terms, *support*, *plausibility*, *singleton objects*, and *mass value* are used in terms of the Dempster-Shafer theory of evidence. Brief definitions of these terms can be found in Appendix A. For each piece of information, the intensity encodes a numeric value with darker regions representing higher numbers.

- The top left quadrant displays all the regions which would answer the question "What are all the regions initially hypothesised as object X?", where X represents the object under consideration. An object is considered as an inital hypothesis if it is contained in the highest ranking subset of the final mass function.
- The top right quadrant shows the mass value assigned to the highest ranked subset for those regions displayed in the top left quadrant.
- The bottom left shows, for every region in the image, the support value for the set containing the singleton object under consideration. The support is computed from the final mass function and can be thought of as the lower bound on the system's belief that this region is this object.
- Finally, the bottom right displays the plausibility value for the set containing the singleton object under consideration. Again, the plausibility is computed from the final mass function and can be thought of as the upper bound on the system's belief that this region is this object.

In figure 6.7, the combined conflict obtained for each region is displayed. A high conflict value (i.e. a dark region) indicates a large amount of disagreement between separate pieces of evidence. This is the value for k contained in the normalization constant of the combination function (see formula 3.2). The conflict measure can be used to pinpoint areas which may need more verification. Also of interest is the conflict obtained by each intermediate combination. This is displayed from left to right, top to bottom as conflict from color, intensity, texture and location respectively in figure 6.8.

7. Conclusion

We have presented here a method which automatically generates a knowledge base for the formation of initial object hypotheses using statistical information provided from a set of training objects. The plausibilty formalism uses a computationally efficient approach to the Dempster-Shafer formalism for representing and combining evidence. The Dempster-Shafer formalism is used for its rich evidential representation. Our system addresses the problems of using heuristic methods for constructing a knowledge base and combining evidence as well as the problems of a strict Bayesian approch. Presented is a set of results showing our plausibility theory applied to a set of color outdoor road scenes which shows that the approach has significant potential.

8. Acknowledgements

We would to thank Edward Riseman and Allen Hanson for their guidance and careful reading of this paper. In addition thanks go to Deborah Strahman for sharing her breadth of knowledge in the areas of probable, possible, and plausible reasoning.

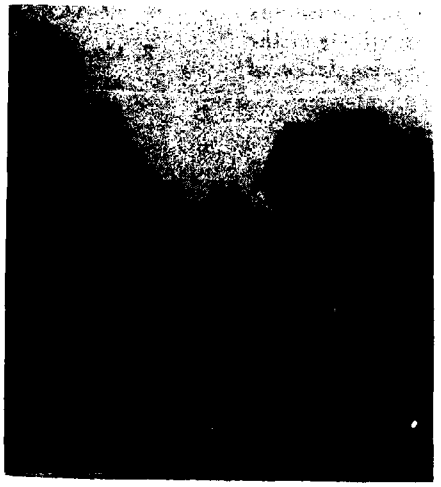


Figure 6.1: Intensity Image
The intensity image digitized to a resolution of 256 by 256. The intensity was computed from the red, green, and blue planes using the formula $R + G + B/3$.

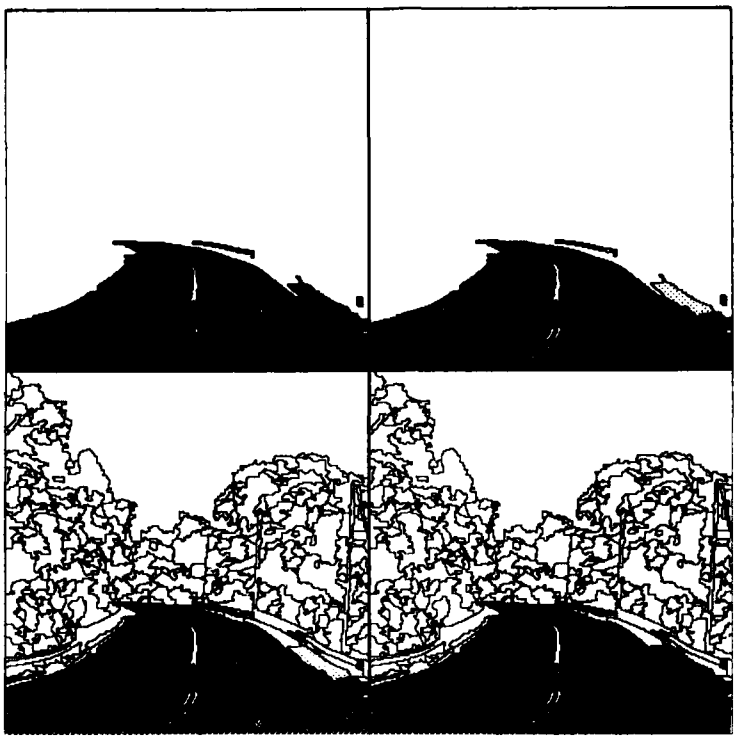


Figure 6.2: Initial Hypotheses for Road

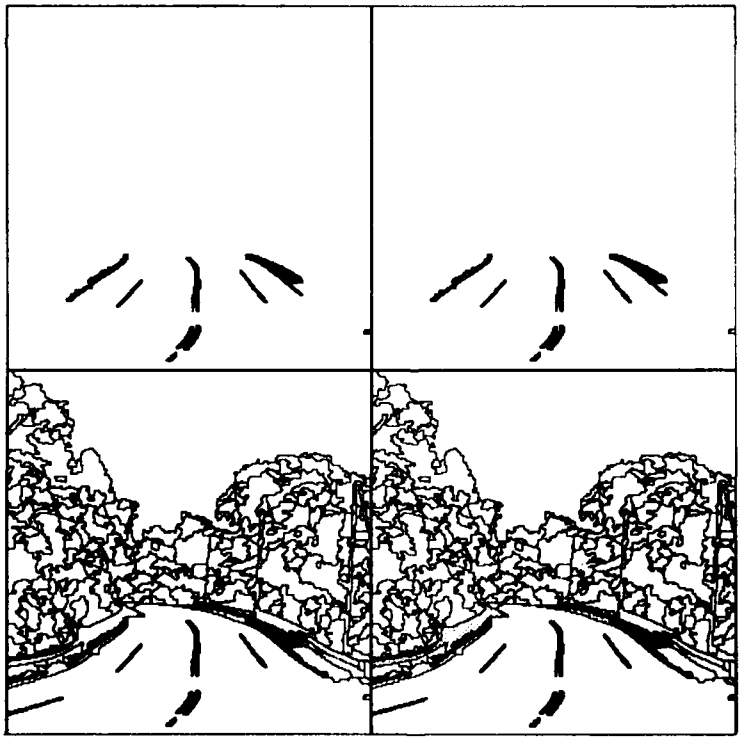


Figure 6.3: Initial Hypotheses for Road Line

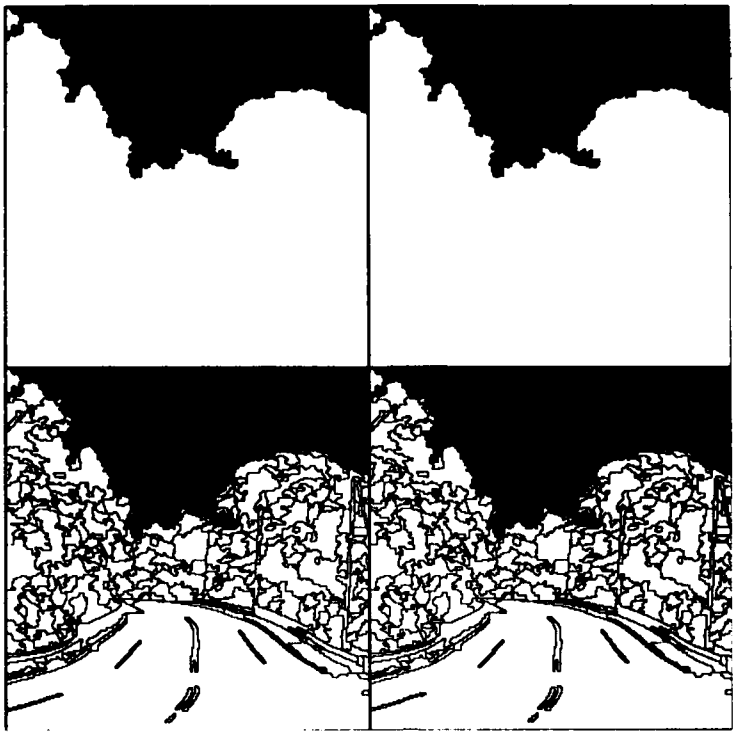


Figure 6.6: Initial Hypotheses for Sky

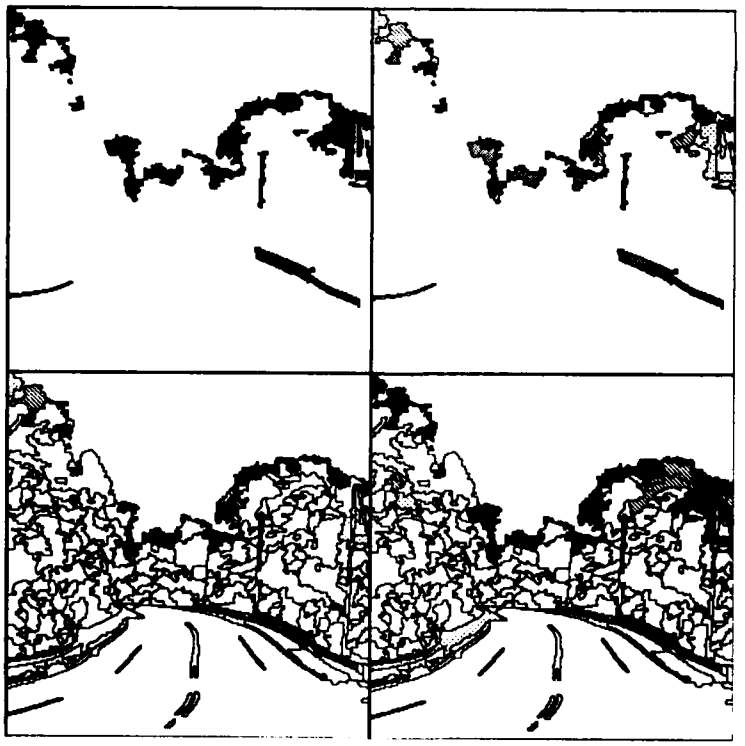


Figure 6.5: Initial Hypotheses for Sky-Tree

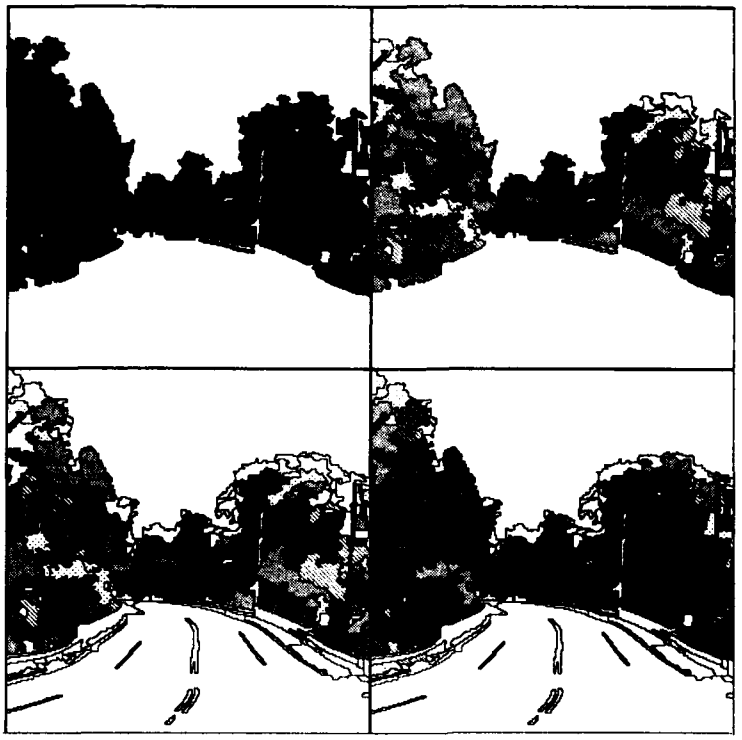


Figure 6.4: Initial Hypotheses for Foliage

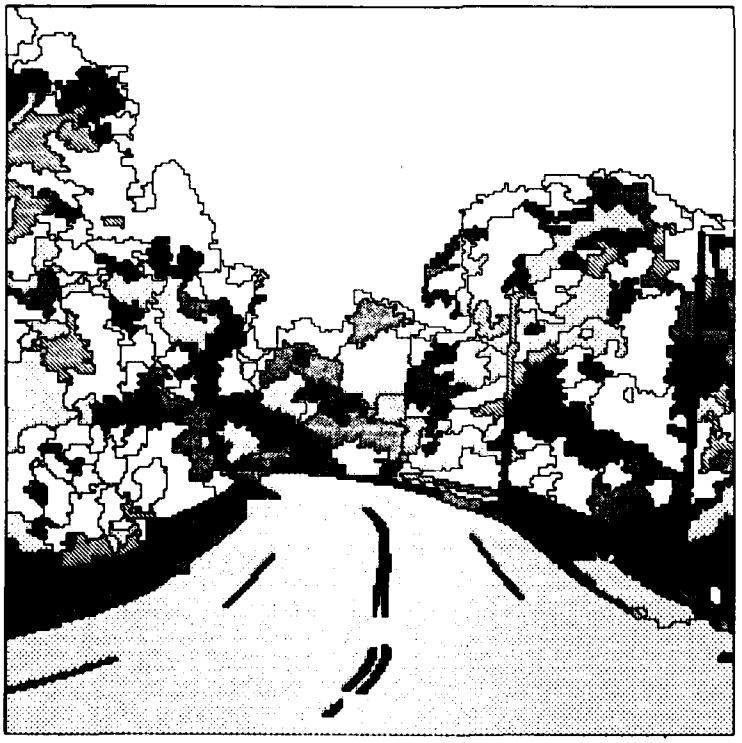


Figure 6.7: Total Combined Conflict

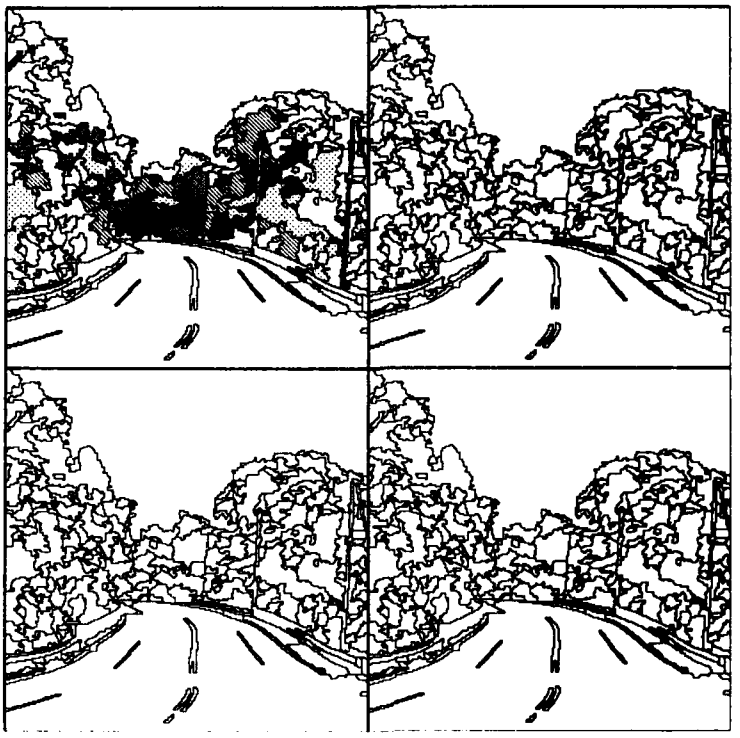


Figure 6.8: Conflict for Color, Intensity, Texture, and Location

A. Dempster-Shafer Tutorial

A.1 Dempster-Shafer — An Intuitive Approach

The Dempster-Shafer theory of evidence provides a rich set of semantics for representing and combining evidence and can be viewed as a least commitment approach. Each piece of evidence is intended to reduce a set of possible hypotheses. The consensus of many pieces of evidence is then used to focus on the smallest possible set of hypotheses. Dempster's combination rule is the heart of the system and much literature can be found describing its focusing methods [Shaf76,Lu84,Low82,Wes84,Rey86a,Rey86b].

A.2 Terms and Concepts

The Dempster-Shafer theory of evidence can be broken up into several concepts:

Frame Of Discernment: A framework which defines a set of possible hypotheses, or possible answers to a given question. The frame of discernment defines a set of assertions: *This is an X*. where *X* is an element of the frame of discernment. A frame of discernment is defined to contain all possible answers to the question to be answered. This assumption can be fulfilled by the addition of some "unknown" object which represents all answers not explicitly contained in the frame of discernment. The set which contains all elements of the frame of discernment is represented by Θ.

A Mass Function: A structure for representing evidence which maps a unit of belief over the powerset of the frame of discernment. The mass value assigned to a subset of the frame of discernment quantifies how much the belief is contained in the system that *exactly* that subset of hypotheses is the best answer to the question.

A Feature Space: Each feature space is fashioned to provide information needed to answer some particular aspect of the given question. A set of feature spaces are designed to bring together many aspects of the evidence needed for an inference.

A Knowledge Source: Maps a feature value for a specified feature space into a representation suitable for combining with other evidence given a combination rule. In this case, a knowledge source maps a feature value into a mass function to be combined with other pieces of evidence using Dempster's Rule.

Dempster's Rule: Dempster's Rule combines evidence as represented by two mass functions to create a consensus of opinion. Dempster's Rule is both associative and commutative providing means for several pieces of information to brought together as combined evidence. Mathematically Dempster's Rule is the orthogonal sum of two mass functions and is defined as follows: For every $C \neq \theta$

$$m_1 \bigoplus m_2(C) = \frac{\sum_{A \cap B = C} m_1(A) \cdot m_2(B)}{1 - k} \quad \text{(A.1)}$$

where

$$k = \sum_{A \cap B = \emptyset} m_1(A) \cdot m_2(B). \quad \text{(A.2)}$$

k is precisely a statement about how much disagreement exits between the two pieces of evidence.

Support, Plausibility, Doubt, and Conflict: Metrics which describe the degree to which some member or some subset of the frame of discernment is the correct answer to a given question.

In the Dempster-Shafer approach, the object space is defined over the powerset of all possible objects. A hypothesis then is not merely an element of the frame of discernment, but rather a set of objects. Each knowledge source is responsible for distributing a unit of belief across the subsets of the frame of discernment. The resulting list of object subsets and mass values is a *mass function*. Higher mass values are associated with more likely hypotheses. Three restrictions apply to mass functions; each subset receives a mass value between zero and one, the empty set receives zero mass, and the sum of the mass values over the object space equals one. For a complete discussion of mass functions see [Shafer].

In general, an inference problem consists of a question to be answered, a frame of discernment containing all possible answers to this question, and a set of feature spaces each designed to provided some piece of information useful in answering the question at hand. This collection is termed a *context*.

A.3 Specific attributes of the Shafer Dempster Approach

As well as providing a combination rule for bringing together disparate sources of evidence, the Dempster-Shafer approach provides explicit information about supporting evidence, plausible evidence, uncertainty of a decision, conflict between knowledge sources, disbelief and no belief. The following subsections describe these types of information.

A.3.1 Supports, Doubt, and Plausibility

The mass values associated with each subset in the object space is only a small portion of the information supplied by a mass function. Of initial importance is the information obtained by asking the following questions for a given subset A: "What amount of of evidence supports A as being the correct answer?", "What amount of evidence refutes A as being the correct answer?" and "What amount of evidence fails to refute A as being the correct answer?". The values obtained by asking these questions are the referred to as *support*, *doubt* and *plausibility* respectively.

$$Spt(A) = \sum_{s \subseteq A} m(s).$$

$$Dbt(A) = \sum_{s \cap A = \emptyset} m(s).$$

$$Pls(A) = \sum_{s \cap A \neq \emptyset} m(s) = 1 - Spt(\neg A).$$

Figure A.1 shows graphically the relationship between the support, doubt and plausibility.

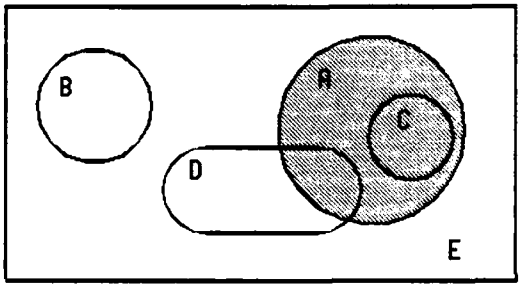


Figure A.1: Venn diagram of support, plausibility and doubt.

$$Spt(A) = m(A) + m(C).$$
$$Pls(A) = m(A) + m(C) + m(D) + m(E).$$
$$Dbt(A) = m(B).$$

A.3.2 Knowledge about Conflict

Implicit with each combined mass function is a statement about how the knowledge sources involved in the decision process agree with one another. This is known as the conflict value and is precisely defined by Dempster's Rule as:

$$k = \sum_{A \cap B = \emptyset} m_1(A) \cdot m_2(B). \quad \text{(A.3)}$$

One assumption mentioned earlier is that all possible objects are represented in the frame of discernment. This may be accomplished by the addition of the *unknown* object in the frame of discernment. If all knowledge sources are required to give some mass to the *unknown* object, then $k = 0$ for all combinations of mass functions and the conflict value is precisely the amount of mass assigned to this *unknown* object.

A.3.3 Disbelief versus No Belief

Disbelief and no belief can be discussed both in terms of a mass function as a whole and about a particular subset receiving mass within a mass function. Looking at a mass function as a whole, statements about support and plausibility are direct statements about disbelief and no belief. No belief is represented as the support for some subset being near zero, where as disbelief is represented as a plausibility near zero. Looking at the mass values themselves. If for a given subset A the mass value is near one, then for any $a \subseteq A$ the system provides little information (no belief) for the support of a but much information about its disbelief for any $b \not\subseteq A$. Disbelief about the systems performance in general (or about a knowledge source) is contained in the appropriate conflict value.

Bibliography

[Binf82] T. Binford, *Survey of Model Based Image Analysis Systems*, International Journal of Robotics Research 1 (1982), pp. 18-64.

[Belk86] R. Belknap, E. Riseman and A. Hanson (1985), "The Information Fusion Problem and Rule-Based Hypotheses Applied to Complex Aggretations of Image Events." *Proceedings: DARPA Image Understanding Workshop*, December 1985, pp. 279-292.

[Burn86] J. Brian Burns, Allen R. Hanson & Edward M. Riseman. "Extracting Straight Lines", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, PAMI-8, #4 (July 1986), pp. 425-455.

[Demp67] A. P. Dempster (1967), "Upper and lower probabilities induced by a multivalued mapping", *Annals of Mathematical Statistics*, vol. 38, 1967, pp. 325-339.

[Demp68] A. P. Dempster (1968), "A generalization of Bayesian inference", *Journal of the Royal Statistical Society*, Series B, vol. 30, 1968, pp. 205-247.

[Demp84] A. P. Dempster and A. Kong (1984), "Belief functions and communications networks", Report, Department of Statistics, Harvard University, Cambridge, MA, Nov. 1984.

[Drap86] Bruce A. Draper, Allen R. Hanson & Edward M. Riseman (1986), "A Software Environment for High Level Vision". *Proceedings AAAI Workshop on Knowledge Based Systems and Tools*, Columbus, Ohio; October 1986.

[Duda73] R. Duda and P. Hart (1973), *Pattern Classification and Scene Analysis*. A Wiley-Interscience Publication, New York. 1973.

[Faug82] O. D. Faugeras (1982), "Relaxation labeling and evidence gathering", *Proc. Conf. Pattern Recognition and Image Processing*, Las Vegas, June 1982, pp. 672-677.

[Hans78] Hanson, Allen R. & Riseman, Edward M. "VISIONS: A Computer System for Interpreting Scenes", in *Computer Vision Systems*, Hanson & Riseman, eds., Academic Press, N.Y., 1978. pp. 303-333.

[Hans86] Hanson, Allen R. & Riseman, Edward M. (1986), "A methodology for the development of general knowledge-based vision systems", in *Vision, Brain, and Cooperative Computation* (M. Arbib and A. Hanson Eds.) to be published by MIT Press 1987: Also COINS Technical Report 86-27, University of Massachusetts at Amherst, July 1986

[Hans87] Hanson, Allen R. & Riseman, Edward M., "The VISIONS Image Understanding System - 86", in *Advances in Computer Vision*, C. Brown (Ed.), Erlbaum, 1987.

[Humm83] R. A. Hummel and S. W. Zucker (1983), "On the foundations of relaxation labeling processes", *IEEE Trans. Patt. Anal. Mach. Intel.*, vol. PAMI-5, no. 3, May 1983, pp. 267-286.

[Kitc80] L. J. Kitchen (1980), "Relaxation applied to matching quantitative relational structures", *IEEE Trans. Syst. Man Cybern.*, vol. SMC-10, 1980, pp. 96-101.

[Kitc84] L. J. Kitchen and A. Rosenfeld (1984), "Scene analysis using region-based constraint filtering", *Pattern Recognition*, vol. 17, no. 2, 1984, pp. 189-203.

[Kohl84] Ralf R. Kohler. *Integrating Non-Semantic Knowledge into Image Segmentation Processes*. Ph.D. thesis, COINS, Univ. of Massachusetts, Amherst, 1984.

[Kybu84] H. E. Kyburg (1984), "Bayesian and non-Bayesian evidential updating", Tech. Report 139, Department of Computer Science, University of Rochester, Rochester NY, July 1984.

[Low82] J. D. Lowrance (1982), "Dependency-graph models of evidential support", Ph.D. Thesis, University of Massachusetts, Amherst, 1982.

[Low83] J. D. Lowrance, T. D. Garvey (1983), "Evidential reasoning: an implementation for multisensor integration", Tech Note 307, SRI Artificial Intelligence Center, December 1983.

[Lu84] S. Y. Lu, H. E. Stephanou (1984), "A set-theoretic framework for the processing of uncertain knowledge", AAAI-84, pp. 216-221.

[Rey86a] G. Reynolds, D. Strahman and N. Lehrer (1985), "Converting Feature Values to Evidence", *Proceedings: DARPA Image Understanding Workshop*, December 1985, pp. 331-339.

[Rey86b] G. Reynolds, D. Strahman, N. Lehrer and L. Kitchen (1986), "Plausible Reasoning and the Theory of Evidence." University of Massachusetts at Amherst COINS Technical Report 86-11, April 1986.

[Ros76] A. Rosenfeld, R. A. Hummel and S. W. Zucker (1976), "Scene labeling by relaxation operations", *IEEE Trans. Syst. Man Cybern.*, vol. SMC-6, 1976, pp. 420-433.

[Shaf73] G. Shafer (1973), "A theory of statistical evidence", pp. 365-434 in *Foundations of Probability Theory, Statistical Inference, and Statistical Theories of Science*, W. L. Harper and C. A. Hooker eds, vol. II.

[Shaf76] G. Shafer (1976), *A Mathematical Theory of Evidence*, Princeton University Press, 1976.

[Shaf84] G. Shafer (1984), "Probability judgement in artificial intellegence and expert systems", Working Paper 165, School of Business, The University of Kansas, Lawrence, Kansas, Dec. 1984.

[Shaf85] G. Shafer (1985), "Belief functions and possibility measures", *The Analysis of Fuzzy Information*, vol. 1, J. C. Bezdek, ed., CRC Press.

[Shaf83] G. Shafer and A. Tversky (1983), "Weighing evidence: the design and comparison of probability thought experiments", *75th Anniversary Colloquium Series*, Harvard Business School.

[Shor76] Shortliffe, E. H. (1976) *Computer-Based Medical Consultations: MYCIN*. New York: American Elsevier.

[Smit61] C. A. B. Smith (1961), "Consistency in statisical inference and decision", *Journal of Royal Statisical Society*, Series B, vol 23, 1961, pp. 1-37.

[Smit65] C. A. B. Smith (1965), "Personal probability and statistical analysis", *Journal of the Royal Statisical Society*, Series B, vol. 128, 1965, pp. 469-499.

[Strt84] T. Strat (1984), "Continuous belief functions for evidential reasoning", AAAI-84, pp. 308-313.

[Wal72] Waltz, D. (1972) "Generating Semantic Descriptions from Drawings of Scenes with Shadows." AI-TR-271. Project MAC, Massachusetts Insititute of Technology. (Reprinted in P. Winston (Ed.). (1975) *The Psycology of Computer Vision*. McGraw-Hill, New York, 19-92.)

[Weis86] Weiss, R. and M. Boldt. (1986) "Geometric Grouping Applied to Straight Lines." Computer Vision and Pattern Recognition, 1986.

[Wes83] L. Wesley (1983), "Reasoning about control: the investigation of an evidential approach", *Proc. Eighth International Joint Conference on Artificial Intelligence*, Karlsruhe, West Germany, August 1983, pp. 203-210.

[Wes84] L. Wesley (1984), "Reasoning about control: an evidential approach", Tech. Report 324, SRI Artificial Intelligence Center, 1984.

[Wes85] L. Wesley (1985), Ph.D. Thesis, University of Massachusetts, Amherst, *in preparation*.

[Weym86] Terry E. Weymouth. *Using Object Descriptions in a Schema Network for Machine Vision*, Technical Report 86-24, COINS dept., Univ. of Massachusetts, May 1986.

[Wood78] Woods, W. A. "Theory Formation and Control in a Speech Understanding System with Extrapolations Toward Vision", in *Computer Visions Systems*, Hanson and Riseman, eds., Academic Press, N.Y., 1978. pp 379-390.

[Zad83] L. A. Zadeh (1983), "The role of fuzzy logic in the management of uncertainty in expert systems", *Fuzzy Sets and Systems*, vol. 11, 1983, pp. 199-227.

GOAL-DIRECTED CONTROL OF LOW-LEVEL PROCESSES FOR IMAGE INTERPRETATION

Charles A. Kohl Allen R. Hanson
Edward M. Riseman

Department of Computer and Information Science
University of Massachusetts
Amherst, Massachusetts 01003

ABSTRACT

The task of image interpretation is viewed as a coordinated process in which high-level interpretation processes and low-level segmentation processes interact through what is known as the *intermediate-level* of processing. Control processes at this intermediate-level respond to requests from the interpretation processes which are expressed in terms of *goals*. The presentation of a request for low-level processing, which is represented by the posting of a goal, results in the creation of an intermediate-level control process which is an instantiation of a control structure known as a *schema*. The set of schemas represented at the intermediate-level define the types of goals which may be processed at this level.

The instantiations of the schemas utilize knowledge of the image domain to translate the goals into appropriate low-level process specifications which include the identification of the image features, algorithms, sensitivity settings, or rules to be used by the process to accomplish the desired task. These low-level process specifications are then passed to a low-level process controller which directs the execution of the process and then returns the results to the intermediate level control process.

Using this control paradigm, high-level interpretation processes gain the capability to create or refine segmentation data according to predefined goals and/or current hypotheses regarding the content of the image, and thereby improve the quality of the overall interpretation.

1. THE INTERMEDIATE LEVEL OF IMAGE INTERPRETATION

Image interpretation is a complex process through which a numeric array, representing a digitized visual scene, can be analyzed to provide a semantic description of the scene content. One subgoal of the interpretation process is **Image segmentation**, the low-level process by which the digitized image is abstracted into a set of primitive elements which may be used by interpretation processes as the basis for the construction of an abstract symbolic model of the original scene. According to many commonly accepted views of interpretation, image segmentation is simply the first stage of the interpretation process. We take the view, however, that segmentation is a process which does not exist in isolation, but rather is an integral part of the overall image interpretation process.

This concept of the interdependence between the segmentation and interpretation processes has developed slowly over the years as researchers have become aware of the difficulty of each of the phases of processing. Researchers such as Marr ([20]), Brooks and Binford ([3]), Thompson ([34]), Reynolds ([29]), Hwang ([15]) and Hanson and Riseman ([30]) have all stressed the need for interaction between high and low level processes.

To date, however, little has been done to address this problem, and thus there are no commonly accepted protocols for the interaction. If we are to adequately deal with interaction between high and low-level processes, there are three separate issues which must be resolved. First, there must be a common knowledge representation which is applicable to all levels of the interpretation process; intercommunication between processes at different levels is difficult or impossible if data objects and hypotheses are not consistent across the (somewhat artificial) boundaries between levels. Second, there is the necessity for a single unified control mechanism which can coordinate and schedule the activities of high and low-level processing tasks. Finally, there is the requirement for an explicit representation of the knowledge necessary for the effective application of low-level processes (i.e. the selection of the appropriate algorithms, sensitivity settings, and image features). For example, we require a mechanism which can encode the knowledge which indicates that certain image features and algorithms are generally useful when segmenting textured image areas, while others are more useful when segmenting smooth image areas.

In this report, we outline the design of a system which has been developed within the VISIONS Image Understanding System ([17]) to deal with these three issues and accomplish the integration of the the high and low levels of the image understanding process. The system is designed to provide the mechanism by which high-level requests for data may activate a set of appropriate low-level processes which may be capable of producing the desired data. The specification of the data request is in the form of a *goal*, a data structure which defines the nature of the image abstractions desired by the requesting process. Attributes stored within the goal data structure, known

*This work was supported in part by the Air Force Office of Scientific Research Grant AFOSR-86-0021, the National Science Foundation Grant DCR-8318776, and DARPA Contract DACA76-85-C-0008

as *constraints*, express additional information which may be useful in the specification of the most appropriate low-level processes. The goal constraints express the desired characteristics of the data to be produced, and may be represented in either semantic or syntactic terms. The semantic constraints are defined in terms of semantic labels (e.g. *"segment a specific portion of the image to separate tree and sky"*), while the syntactic constraints are expressed in terms of measurable image features (e.g. *"produce regions which exhibit homogeneous texture measures"*). Through the use of the information expressed in these constraints, the system is able to select the most appropriate image features, algorithms, sensitivity settings, rules, etc., for the specification of the low-level task.

In VISIONS, the interface between the high and low levels of the interpretation process is called the *intermediate* level; data at this level is represented by *tokens*. Tokens may represent any abstraction of the raw image data which is in spatial registration with the actual data and/or groups of these abstractions which have been formed by the application of grouping operations ([35,2,28]). Thus, intermediate level tokens may represent any of the normal types of segmentation output, such a regions, lines, or surfaces. The intermediate level is distinguished from the low level in that there is no explicit representation of pixels, and from the high level by the requirement that tokens are in spatial registration with the image data.

The GOLDIE (Goal Directed Intermediate-Level Exectutive) system ([16]) has been developed as a mechanism to provide this intermediate-level control of low-level processing. The basic control structure of GOLDIE is the *schema*, a declarative specification of *control* strategies which may be used by the system to satisfy a specific goal. Schemas provide a flexible and extensible control structure which is used within VISIONS to direct both the high and intermediate processing levels ([36,27,11]), and they provide a natural interface between processes at each of these levels. GOLDIE extends this concept of schema-directed control to the low-level processes.

2. THE NEED FOR GOAL-DIRECTED CONTROL

Low-level segmentation processing is in many ways more of an art than a science. Despite an enormous number of excellent techniques and algorithms (e.g. see [31,14,1,8,10]), no general methods have been found which perform adequately across a wide variety of images. Methods such as region-growing ([12]), histogram cluster labeling ([22]), thresholding ([18]), zero-crossings ([13,20]), and edge and line detection ([5]) have all been found to be effective within restricted domains, but rarely demonstrate the robustness necessary for generalized image interpretation.

The failure of general segmentation techniques to provide "good" segmentations can be traced to a variety of causes. In many cases the image data itself is complex because of the physical situation from which the image was derived and/or the nature of the scene. Heavy textures, unconstrained lighting, view angle, surface reflectivities, markings, and curvature all conspire to produce ambiguous image data. This results in under- or over-merged regions in the region segmentations and missing, fragmented, or incorrectly merged lines in the edge/line segmentations. The type of error often depends upon

Figure 1: Intensity Image of Outdoor Scene

which feature(s) is being used to produce the segmentation and whether the algorithm has a global or local view of the image data during processing. The type of error produced quite often varies spatially within a single image, depending on the type of data in each locality of the image.

To take a specific example, the intensity surface of an image (Figure 1) may contain a great deal of high frequency information which typically will lead to significant *fragmentation* in the segmentation such as shown in Figure 2. Here we see that the region segmentation, which was produced through a zero-crossing algorithm ([20]), has produced a large number of small regions across the textured projection of the tree. While these regions do represent image areas of local homogeneity, in most cases they are of little semantic interest. If our overall interpretation goal is to identify the gross objects in the scene, a segmentation such as that shown in Figure 3, which is produced through a one-dimensional histogram clustering algorithm, would be much more useful in the identification of the trees in the image.

In the case of the smoothly varying or flat surfaces of this image, however, the opposite is true. Here, the slow gradients of the intensity surface result in *missing object boundaries* in the segmentation produced by the clustering algorithm, resulting in the class of segmentation error known as over-merging. An example of this class of error is the lack of a boundary between the sky and house wall in Figure 3.

Existing image interpretation systems attempt to compensate for these two types of error within the high-level interpretation process. Undermerging errors are typically corrected with grouping processes which are used to merge adjacent re-

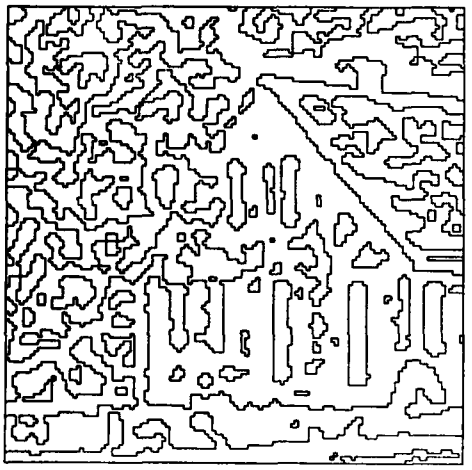


Figure 2: Zero-Crossing Segmentation

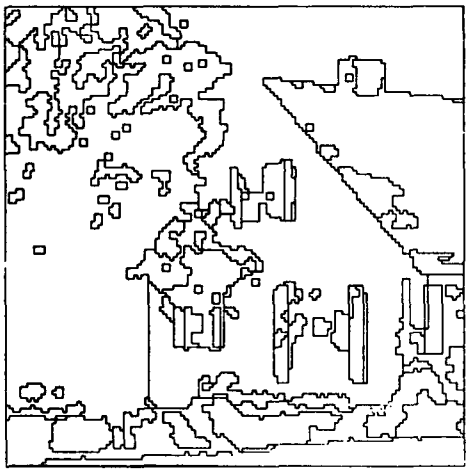


Figure 3: Histogram Clustering Segmentation

gions with similar labels ([37,33]) . Overmerging errors are typically bypassed implicitly through the use of a segmentation which has such a fine resolution that it can be assumed that any object boundary which does exist in the image has a corresponding region boundary in the image ([9,19,25]).

However, since both types of errors may occur simultaneously in different areas of any given segmentation, it is our assertion that error correction properly belongs within the domain of the low-level segmentation processes themselves. As an interpretation process discovers problems with a segmentation, the segmentation itself should be redefined to resolve those problems. This redefinition of the segmentation may require the application of a different segmentation algorithm, or of the same algorithm with different sensitivity settings, over the portion of the image which presents the difficulties for the interpretation process.

Another factor leading to variability in low level segmentation processing has to do with the *choice of image features* over which the segmentation processes will run. In typical RGB images, a large number of computed image features can be defined, some of which are known to be effective in segmentation processing ([24,25,32,21]). However, in a large number of segmentation algorithms, only Intensity (the average of R, G, and B) is used. In other algorithms, the selection of of features is typically made in an *ad hoc* fashion, where the general experience of the person designing the segmentation algorithm is brought into play. If, however, such knowledge and experience were represented explicitly in the system, it would be possible to perform this selection automatically in response to the overall goals of the interpretation process.

Finally, there is the problem of *evaluating* different region/line segmentations. Despite some efforts to quantify the evaluation of segmentation processing ([23,26]), it has been our experience that no low-level evaluation measure restricted to making measurements on the segmentation can provide a useful comparative metric. Rather, the quality of a segmentation can be measured only with respect to the goals of an interpretation process which is applied to the segmentation data. Different criteria must be used to evaluate the quality of segmentation results on highly textured image areas than on smoothly varying image areas.

In view of this variability in the selection of different segmentation methodologies and the difficulties of evaluating the results of various techniques, it becomes apparent that some form of intelligent intermediate-level control is necessary. We believe that the most effective form for this control is a mechanism which can select and apply the most appropriate segmentation process for each distinct area of the image. Through the use of an initial region segmentation of the image, these distinct image areas may be coarsely distinguished, and then through an iterative process of resegmentation and merging (using appropriate algorithms and rules), the areas may be more precisely identified. In other words, our methodology is to form a hierarchical set of segmentation plans in which the results of early processing are combined with the intermediate results of the interpretation process and used to guide and constrain the later processing.

The knowledge required under this control paradigm provides the rules and information by which the goal constraints may be used to determine the utility of the various low-level processes available to the system. In the specification of these low-level processes, the control process is able to choose among a wide array of possible intermediate and low-level tools which include:

- a set of parameterized segmentation algorithms,
- a set of low-level image analysis and enhancement routines,
- a set of measurable image features,

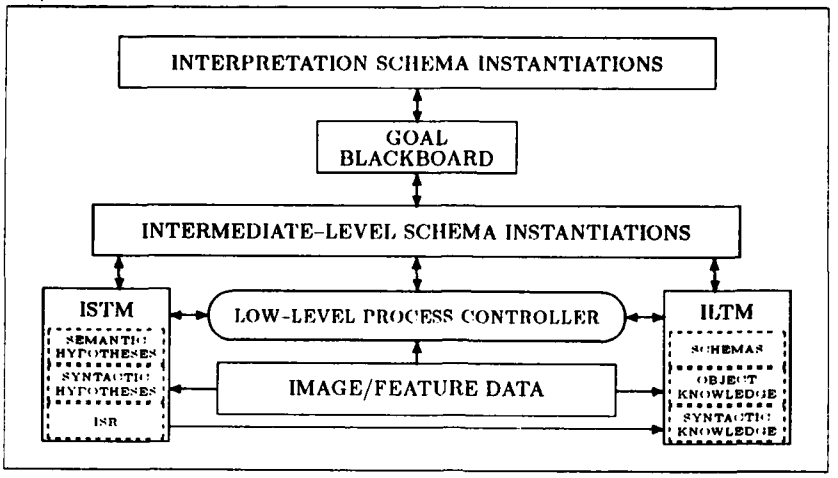


Figure 4: Overview of the GOLDIE System

- processes for accessing and modifying the tokens and their associated attributes in the intermediate-level database, and
- a formalized specification of segmentation goals.

In the next section, we demonstrate the mechanisms by which GOLDIE represents knowledge of these processes and *tools*, and how it uses the knowledge to control low-level processing.

3. INTERMEDIATE-LEVEL KNOWLEDGE AND CONTROL

The GOLDIE system, outlined in Figure 4, may be described in terms of five major functional components: the low-level process controller, the data structures for intermediate-level tokens and hypotheses (ISTM), the representation of explicit intermediate-level knowledge (ILTM), the representation of the control state of the system (Goal Blackboard and Schema Instantiations), and the actual control process which manages the system (represented in the figure by the arcs between the other modules). In this section, we will present a brief description of each of these components, and then demonstrate the mechanisms by which the components are combined to produce a working system.

All of the actual image processing that takes place within this system is managed by the *low-level process controller*. The controller maintains a consistent protocol for the activation of low level processing tasks, permitting the schema representation of low-level processing to be expressed in a manner independent of the actual procedural implementation of these tasks. Thus the addition, deletion, or modification of low-level processing tasks may be accomplished with minimal modification of the schemas which control the tasks. Since many low-level tasks produce intermediate-level data in the form of tokens, the controller is also responsible for the maintenance of a data structure which encodes the relations between tokens and the processes by which they were created. This data structure provides the data necessary for backtracking if the schemas decide at any point that specific segmentation processing should be "undone". Thus if several different attempts were made to segment an area of the image, but only one of these was found to be acceptable, this mechanism allows the deletion of the tokens produced by the unacceptable segmentation processes.

The dynamic *data structures* of the GOLDIE system are the sets of tokens and hypotheses about tokens which are stored in ISTM (Intermediate-level Short Term Memory). The tokens themselves are defined through the ISR (Intermediate Symbolic Representation), a combination database and semantic network in which tokens are defined as graph nodes with associated bitmaps which represent the spatial extent of the token in the image. Attributes of the tokens, such as their compactness measure or mean feature value, are computed on demand by the ISR and then stored in the database for efficient retrieval by any other process which may require the data. Relations between tokens, such as boundary contrast between two adjacent region tokens for a given image feature, are represented as arcs which have values indicating the nature of the relation. Tokens may be directly accessed by name or associatively accessed through the specification of constraints on their attributes.

ILTM (Intermediate-level Long Term Memory) encodes the image-independent *intermediate-level knowledge* structures of GOLDIE which includes schemas, object domain knowledge, and general knowledge of image syntax. These structures form the core of the system in that they contain the knowledge which permits the intermediate-level schemas to utilize low-level processes to achieve high-level goals without explicit high-level control.

As was stated earlier, the schemas of the ILTM are the modular representation of the set of strategies which may be used in an attempt to satisfy the processing requests we call goals. Each schema is designed to serve one particular type of goal, and thus there is a one to one correspondence between the goals that can be recognized by the GOLDIE system and the schemas of the ILTM. The strategies of the schemas specify the various low or intermediate-level tasks through which the goal may potentially be satisfied. The execution of the strategies will typically involve a sequence of operations which may include the posting of subgoals, the execution of image processing tasks through the low-level process controller, and the construction of hypotheses based on data represented in the ISTM. Message-passing constructs allow the executing strategies to communicate and synchronize their activities.

Another knowledge structure of the ILTM is the object domain knowledge which encodes information about the appearance of objects in an image which can be useful in the specification of segmentation tasks. This knowledge is represented as a hierarchical set of graph nodes corresponding to specific semantic objects or classes. Associated with each node is the set of features which have been experimentally shown to best discriminate that object from all other objects in a training set of images. Each of these nodes also contains heuristic information in the form of a syntactic (i.e. feature based) characterization of the object (e.g. textured, smooth, gradient), the segmentation resolution at which the object may be best extracted (high, medium, or low), and a set of rules which can be used by the intermediate-level system to provide crude hypotheses about the possible semantic identity of a particular region token.

ILTM also contains the system's knowledge of image syntax. This knowledge encodes the heuristic information which permits the evaluation of a token, without any use of semantic information, to create a hypothesis as to whether further processing is required. This knowledge include- the set of types of regions (textured, smooth, gradient) or lines (low-contrast, high-contrast, horizontal, long, etc.), and the set of rules to be used to syntactically evaluate each of these types of token. The results of the application of these rules are used to create syntactic hypotheses about the tokens which indicate the degree

Region-Segmentation
'((*region-token* .Region-000042)
(*objective* . object-hypothesis-conflict)
(*evaluation constraint* . nil)
(*object-labels* . (tree grass))

Figure 5: Goal Node Representation

to which the image feature data supports the existence of the token (e.g. a set of rules which are to be used to determine whether a region token which is categorized as textured should be resegmented).

Syntactic knowledge also has to do with the utility of specific image features and algorithms in the creation of tokens with specific characteristics. For example, knowledge structures in the ILTM encode experimentally derived information that indicates that certain features and algorithms are generally useful when segmenting a textured region, while others are useful when segmenting smooth or gradient regions.

The actual control of processing in GOLDIE is managed through another semantic network which represents the *control state* of the system. Goals and schema instantiations are represented as nodes of this network, and arcs represent the relations between these entities ([36,7,6]). The goal blackboard is a section of the network in which goal nodes, representing requests for intermediate-level processing, are created by the schema instance that requires the results of that processing. Any constraints on the request are expressed as attributes of the goal node. By using an attribute-value list to express these constraints (Figure 5), the schema instantiation which is posting the goal can express any information which may possibly be of use to the responding schema instance. The schema instance which responds to a goal will extract the value of any named attribute which it can utilize in processing.

The *control process* of the GOLDIE system continually monitors this goal blackboard for the existence of new goal nodes. As a new goal is observed, the control process creates a new instance of the corresponding schema. The node for this schema instance is linked to the goal node by an arc which represents a contract by the instance to attempt to satisfy the goal. The communication between invoking and invoked schema instances is achieved through a mailbox on the goal node. When the invoked schema instance has obtained results which may satisfy the goal, these results are placed in the mailbox of the goal. The invoking schema process is then able to examine the results and either accept them as satisfying the goal, or request that additional strategies be used in an attempt to produce more acceptable data.

As a way of minimizing the number of unacceptable results, many schemas are capable of using a specific type of constraint known as the evaluation constraint. This constraint is expressed as a function value pair which is used to evaluate results prior to their being returned to the invoking schema instance. If the value returned from the application of the function to a potential set of results does not exceed the threshold value expressed in the constraint, the invoked schema instance will automatically continue processing until a set of results is found which can meet this constraint. This mechanism makes it possible to tune the schema processing to highly specific goals. For example, an evaluation constraint could be created such that the region segmentation schema would return only segmentations which contained regions of a certain shape or color, or both.

A typical scenario illustrating the interaction between GOLDIE and the high-level interpretation system could involve a situation where a high-level interpretation process had formed two conflicting hypotheses regarding the semantic label to be assigned to a particular region token. Under the assumption

that the region token actually represented an area that contained two different objects, the interpretation schema instance would post a goal (Figure 5) for region segmentation. The constraints on the goal would indicate the region token of interest, and would additionally indicate that the reason for the goal posting was that there was an object hypothesis conflict for the two specified object labels.

In response to this goal posting, the system would activate an instance of the region segmentation schema. This schema instance would select image features and algorithms appropriate for the discrimination of the two objects and then initiate a segmentation process which would hopefully split the original region into two new regions, each representing one of the two objects. Tokens corresponding to these two regions would then be returned to the interpretation schema instance for evaluation. If the interpretation schema instance were satisfied with the results, the goal and region segmentation schema instance would be deleted. Otherwise, the schema instance would continue processing until an acceptable segmentation were found or until the strategies of the region segmentation schema instance were exhausted.

4. INTERMEDIATE–LEVEL SCHEMAS

The set of intermediate–level schemas which have been implemented in the GOLDIE system are shown in Table 1 and Figure 6. Although the figure implies a hierarchy of schemas in the system, this hierarchy is designed only to demonstrate the relationship between the initialization schema and the other schemas of the system. We will use this hierarchy as the basis for our discussion, although it is important to recognize that the actual interaction between the schemas is more complex than indicated by Figure 6.

4.1 The Initialization schema

At the highest level of the hierarchy is the **initialization** schema. This schema is typically intended to be invoked at system startup to provide the initial set of image tokens for the interpretation process. Since the interpretation schemas of the high–level system are designed to operate across sets of tokens representing image abstractions, no interpretation processing may take place until we have an initial segmentation produced by the initialization schema.

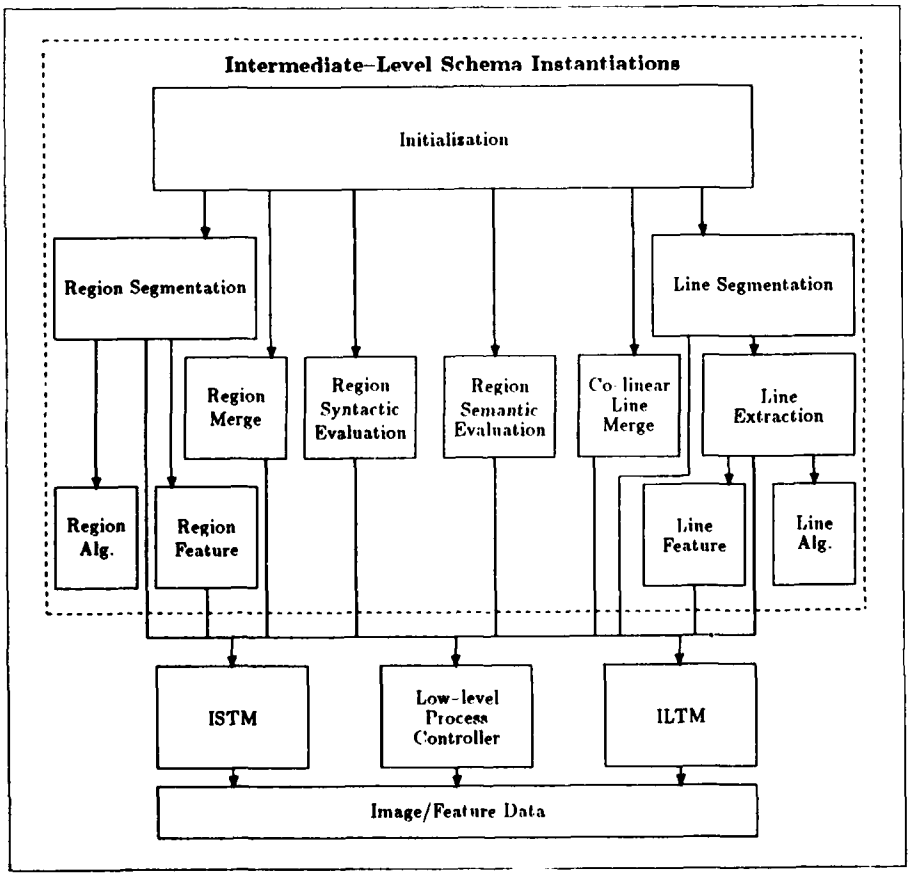


Figure 6: Schemas of the GOLDIE System

Schemas of the GOLDIE System	
Schema Name	Action
initialization	Obtain the "best" segmentation of the specified region token (possibly the whole image) with respect to (possibly initial) specified constraints
region segmentation	Segment a region token
region-algorithm	Select an algorithm for region segmentation
region-feature	Select an image feature for region segmentation
region merge	Merge adjacent region tokens which meet criteria selected as a result of the constraints
syntactic region evaluation	Evaluate region tokens for resegmentation or merging with respect to specified constraints
semantic region evaluation	Provide crude hypotheses for the semantic identity of the specified region tokens
line segmentation	Segment region tokens using evidence from a (set of) line token(s)
line extraction	Extract line tokens from image data
line-algorithm	Select an algorithm for line extraction
line-feature	Select an image feature for line extraction
collinear line merge	Merge adjacent line tokens which meet the criteria selected as a result of the constraints

Table 1: Intermediate-level Schemas Descriptions

The intent of this schema is to produce the "best possible" segmentation of a region token without any assistance from interpretation processes. If used to provide the initial segmentation data, the only constraints on the satisfaction of the initialization goal are the *a priori* constraints of the overall system. For example, if we were implementing a system whose primary goal was to identify different types of trees in outdoor scenes, the *a priori* constraints on the goal for initialization would be "*emphasize regions with texture characteristics*" and "*emphasize tree objects*". In this way, the initial segmentation which is presented to the interpretation system has already been at least partially tuned to the overall goals of the interpretation process.

The initialization schema directly or indirectly makes use of all of the other intermediate-level schemas of the system in this attempt to provide the "best" data-directed segmentation, and is the most complex schema present in the GOLDIE system. Although the schema contains only a single strategy, the execution of this strategy involves region segmentation, region merging, line segmentation, region evaluation, and recursive invocation to achieve the desired result.

The initialization schema requires a region token as input to define the area over which the segmentation is to take place. In the case where the schema is being used to initialize the system, the bitmap for this region token is defined as the entire image. The first step of the strategy is the posting of a goal for region segmentation on this region token. Unless overridden by its own goal constraints, the initialization schema instance will specify an evaluation constraint on the region segmentation goal which makes use of the syntactic evaluation hypotheses created by an instance of the syntactic region evaluation schema. The function associated with this evaluation constraint requires that the majority of the region tokens produced are hypothesized to be "good". In this context, "goodness" would indicate that no further segmentation was necessary, and thus the function computes a value based in part on the resegmentation hypotheses associated with each of the region tokens in the segmentation. Obviously, if this were the only factor, the best segmentation would be one in which each pixel were represented as an individual region token, and thus other factors such as the number of region tokens are also used in this function. The specification of this constraint is an attempt to balance the cost of further segmentation against the cost of region merging to produce the highest quality segmentation at the lowest computational cost. Figure 7 shows the segmentation selected by the region segmentation schema according to this evaluation constraint for the image from Figure 1.

Once such a segmentation has been obtained, it is typically the case that despite good overall quality, the segmentation data may still contain many obvious (to an observer) errors which can be corrected at the intermediate level. One such

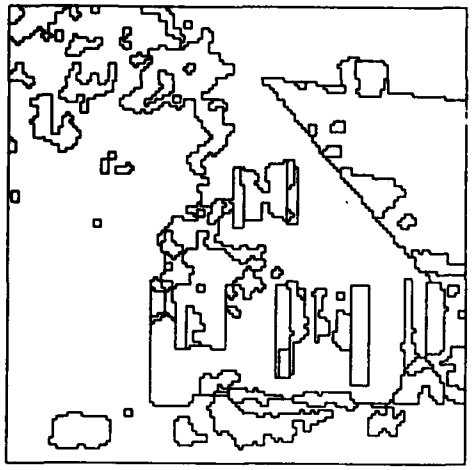


Figure 7: Top–Level Segmentation from Initiallization Schema

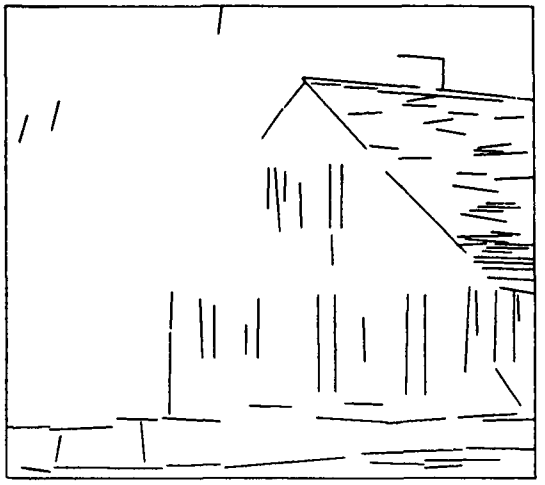


Figure 8: Long High Contrast Lines

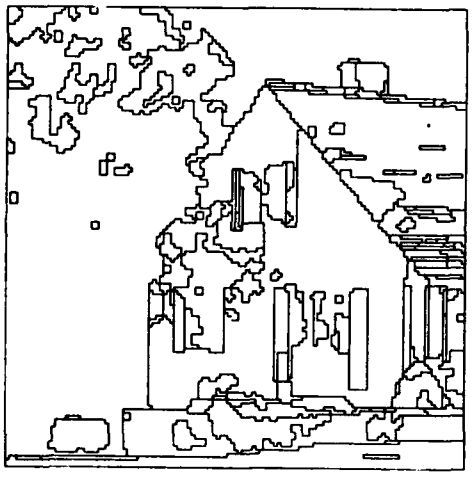


Figure 9: Insertion of Long High Contrast Lines

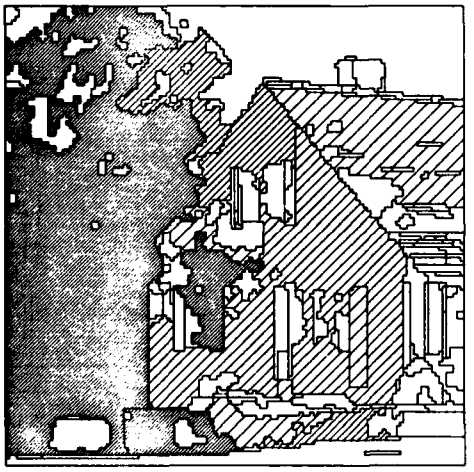


Figure 10: Regions with Strong Resegmentation Hypotheses

category of error is the possible absence of significant long lines which are obvious in the raw data. Thus the next step in the initialization strategy is to post a line segmentation goal for the insertion of all long, high contrast lines. In this process, which is somewhat similar to the process employed by Nazif ([23]), a set of line tokens are extracted from the raw data by an instance of the line extraction schema and the long, high contrast lines (Figure 8) are used to split any existing region tokens that they intersect (Figure 9).

The execution of this stage of the strategy attempts to assure that all significant discontinuities in the raw image data are represented by the boundaries of region tokens in the segmentation, but fails to assure that all diffuse object boundaries (i.e. those represented by slow spatial changes in feature values) will be represented. Therefore, long line insertion is followed by the posting of a goal for the syntactic evaluation of the region tokens. Through the execution of the syntactic region evaluation schema, it is expected that when a given region token spans such a diffuse boundary, the evaluation rules used by the schema will produce a strong resegmentation hypothesis. Figure 10 indicates, with cross–hatching, those regions for which the resegmentation hypotheses are strong enough to indicate a need for resegmentation. The stronger the hypothesis, the more dense the cross–hatching.

Using the goal mechanism of the system to recursively invoke the initialization schema over the set of region tokens for which such a hypothesis exists, the segmentation may be refined in the image areas which contain these diffuse boundaries.

Since the image data is progressively more localized as the system proceeds with this recursive invocation, more use may be made of local context to focus on the correct algorithms and features for segmentation. For example, by using semantic hypotheses produced by an instance of the semantic evaluation schema, the instances of the initialization schema are able constrain the instances of the region segmentation schema which they invoke. Thus, if an initialization schema were invoked over a region token which spanned the boundary between bush and tree (as indicated in Figure 11), the constraining knowledge that both semantic objects might be present in the region could help the associated region segmentation schema instance to select features and algorithms which could best discriminate these two objects. Figure 12 demonstrates the nature of this process.

In this case, the overmerged region on the left of Figure 10 is resegmented using the hypotheses for the existence of both bush and tree in the region. Based on this knowledge, the image feature which is selected for the segmentation task is a rotation of RGB space which maximally distinguishes the mean RGB of bush objects from that of tree objects (according to data which has been extracted from a training set and stored in ILTM). With respect to the original image in Figure 1, it can be seen that the segmentation in Figure 12 has distinguished tree from bush and has captured all of the highlight/shadow boundaries in the tree. However, the segmentation is far from ideal in that the two different types of bush in the bottom left have not been distinguished, and also in the fact that many of the tree highlight/shadow boundaries are quite local and semantically unimportant. But since the area defined by the original tree/bush region (Figure 11) is being processed by a complete recursive instance of the initialization schema, the remaining tasks of the initialization strategy, including long, high contrast line insertion, potential recursive invocation of

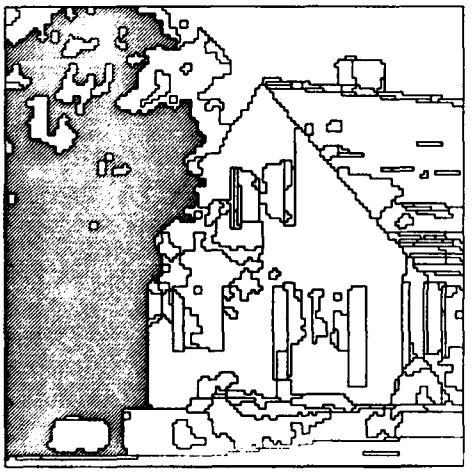


Figure 11: Tree/Bush Region

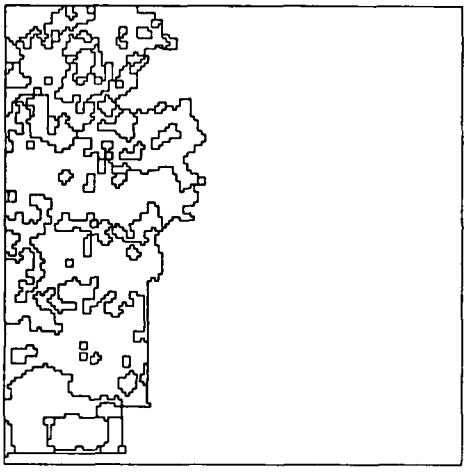


Figure 12: Resegmentation of Tree/Bush Region

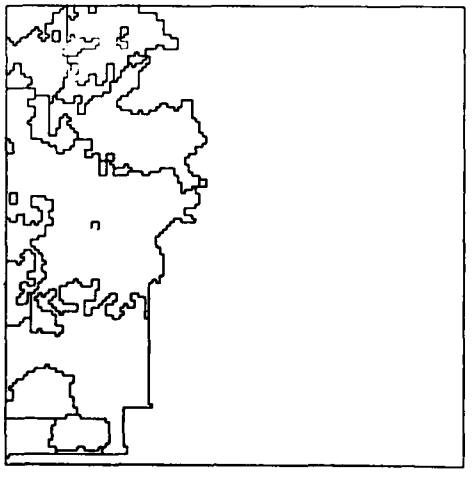


Figure 13: Final Result from Resegmentation of Tree/Bush Region

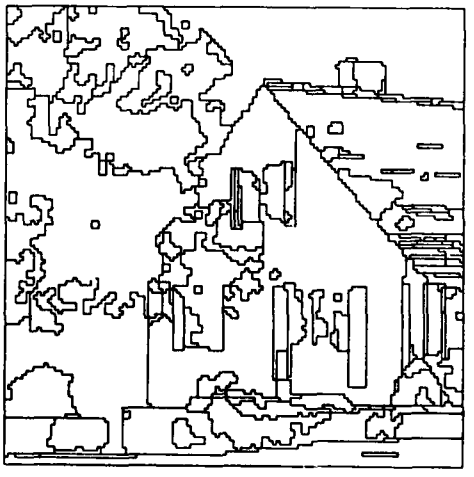


Figure 14: Initialisation Schema Segmentation Prior to Final Merge

the initialization schema, and region merging (see below) are used to complete the processing on this area. The final result which is returned by this instance of the initialization schema (Figure 13) identifies all major semantic boundaries in the area without a great deal of fragmentation. Note that several of the small regions near the top of the image which appear to be artifacts of fragmentation were not defined as being part of the resegmentation region, and thus may not be merged by this particular instance of the initialization schema.

At the point that all recursive invocations of the initialization schema have completed, and all region tokens in the ISR have acceptably low resegmentation hypotheses, experience has shown that the updated segmentation typically will still display some degree of overfragmentation (Figure 14). The region merge schema is therefore invoked by the initialization schema to merge all adjacent regions which exhibit reasonable similarity under the constraints specified.

Since the constraints specified in the original instance of the initialization schema are passed down into any goals posted by that schema instance, this entire process occurs within the original context. Thus, even though a complex chain of recursive invocations of the initialization schema has occurred, the top level constraints are observed at all levels of processing.

The final segmentation results produced by the initialization schema (with no *a priori* constraints) on the image from Figure 1 are shown in Figures 15 and 16. This segmentation appears to be qualitatively more useful than either of the two earlier segmentations which were produced by segmentation algorithms that were applied globally and uniformly across the image. The types of errors which do remain (e.g. overfragmentation near the gutter of the house, and overmerging in the windows) are generally the result of ambiguous image data and would require high-level interpretation processing to resolve the ambiguity or direct the additional reorganization of the intermediate-level tokens.

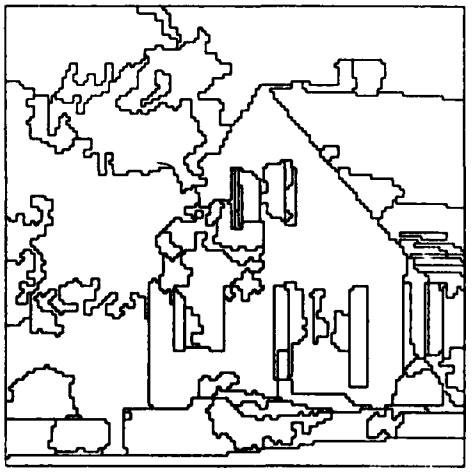


Figure 15: Segmentation from Initialization Schema

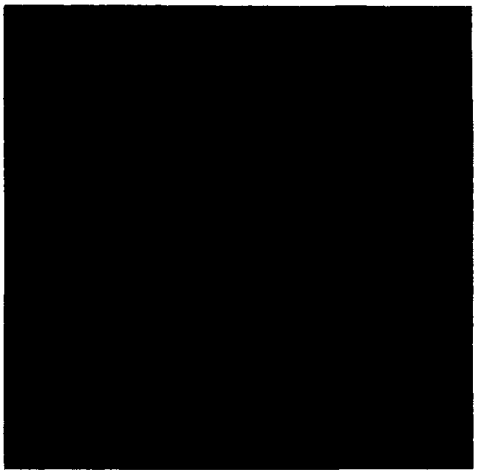


Figure 16: Image with Overlaid Segmentation from Initialization Schema

4.2 The region schemas

The intent of the **region segmentation** schema is to produce a high quality segmentation over the portion of the image specified by a region token. If explicit constraints are known, such as *"concentrate on regions of the image which have the potential for being interpreted as tree"*, the schema is able to use strategies which constrain the segmentation processes such that the tokens produced should meet the expectations of the invoking higher level schema. In the case of this particular goal constraint, the schema would make use of syntactic knowledge about trees, such as the fact that they are present in textured areas of the image or that they exhibit certain color characteristics, to select the low-level segmentation task which would best discriminate tree from all other objects present in the image.

The region segmentation schema is designed to work according to a hypothesize-and-test paradigm. The schema hypothesizes appropriate processing parameters such as image features, segmentation algorithms, and sensitivity settings, and then uses these parameters to control the application of specific low-level image processing tasks through the low-level process controller. The test portion of the hypothesize and-test paradigm is then implemented by the use of the evaluation constraint associated with the goal. If the test fails, the schema instance continues to hypothesize new features, algorithms, and settings until the test succeeds. If no segmentation is found which meets the evaluation constraint, the segmentation which came closest to meeting the constraint is returned. Figures 17 and 18 show two segmentations for the tree/grass region (Figure 11) which were rejected according to this mechanism prior to the acceptance of the segmentation which was shown in Figure 12. With respect to the evaluation constraint described above, the segmentation in Figure 17 received a low score due to a need for significant resegmentation, and the low

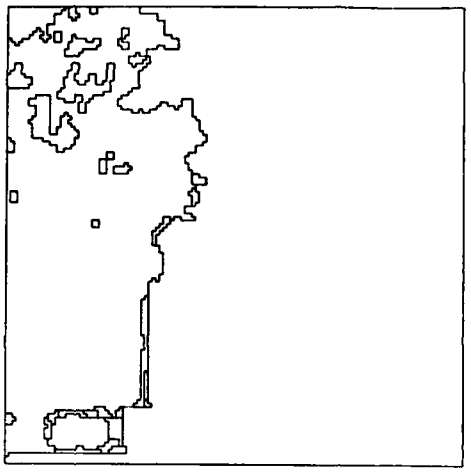


Figure 17: Region Segmentation Which Was Rejected By Region Segmentation Schema

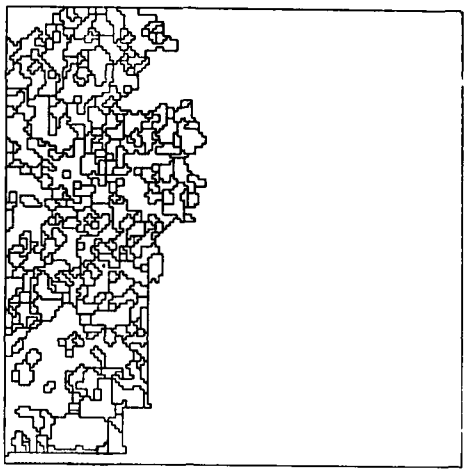


Figure 18: Region Segmentation Which Was Rejected By Region Segmentation Schema

evaluation score for the segmentation in Figure 18 was due to overfragmentation. Note, however, that given a different evaluation constraint, either of these segmentations could have been chosen over that of Figure 12.

Instances of the region segmentation schema make use of region-feature and region-algorithm schemas to establish hypotheses for the parameters involved in the specification of low-level segmentation tasks. Within the **region-feature** schema, initial hypotheses for the selection of an appropriate image feature are made on the basis of the specified constraints. Thus, if the constraints for an instance of this schema specified an interest in tree and sky, the schema would use the knowledge in the ILTM to hypothesize a set of features which would best discriminate regions representing these two objects. The region-feature schema instance would then refine these initial hypotheses by evaluating the actual frequency distribution of each of these features over the area of interest. Based on this evaluation, the hypotheses would be ranked and returned to the invoking schema instance in order of hypothesized utility. If none of this initial set of hypothesized image features were acceptable to the invoking schema, additional less restrictive strategies would then be employed by the region-feature schema instance to propose additional segmentation features.

If, however, no constraints were specified on the goal, the schema instance initially hypothesizes a set of default features which are known to typically produce good results across the particular image domain. It would be this set of features which was evaluated and returned in order of hypothesized utility. Thus we see that, as is the case with most of the schemas in this system, the more specific the information provided on the goal for the region segmentation schema instance, the more specific the hypotheses.

The GOLDIE system currently utilizes a set of 43 image features such as red, green, blue, intensity, local deviation of intensity, hue, etc., all of which are computed from the original RGB. Given the mechanisms by which these features are managed, we believe that it would be a straightforward extension to include other types of spatially distributed data such as infrared or range data.

The **region-algorithm** schema functions in much the same way as the region-feature schema, but in this case the schema instance does not actually initiate any low-level processes. This schema contains a set of strategies for proposing segmentation algorithms and sensitivity settings which are based on the nature of the particular image feature and the other constraints. As currently implemented, this schema is able to select between algorithms for global one-dimensional histogram clustering, localized one-dimensional histogram clustering, zero-crossings, thresholding, and global two-dimensional histogram clustering, each of which can be used with a variety of sensitivity settings. Given a particular image feature, or set of image features, an instance of this schema is able to hypothesize the algorithm and sensitivity setting which is most likely to produce region tokens with the desired characteristics.

The schema first proposes the most specific algorithm that it can hypothesize under the constraints, and if this is not accepted, it can then propose progressively more general algorithms. For example, if an instance of this schema were invoked with the constraints that the region-feature was Deviation (a textural feature which tends to smooth over object boundaries) and the object label of interest was tree, the schema instance would first propose the thresholding algorithm with low sensitivity. This algorithm would be expected to distinguish high-texture areas (tree) from low-texture regions without introducing excessive fragmentation in either area. Additionally, the thresholding algorithm would be expected to place the boundary between region tokens at the midpoint of the image feature gradient of the boundary, hopefully restoring the blurred object boundary. If this algorithm was found to be unacceptable, the schema instance would then propose a low sensitivity zero-

crossing algorithm. If this were also rejected, a final hypothesis of low-sensitivity one-dimensional histogram clustering would be proposed.

Once the region segmentation schema has produced an acceptable set of region tokens, other region schemas may be used to evaluate or modify these tokens. The **syntactic region evaluation** schema is concerned with the creation of ISTM hypotheses about region tokens which indicate a belief in the "goodness" of a particular region structure (i.e. the degree to which the existence of the region token is supported by the underlying image feature data). Based on assumptions expressed in goal constraints, such as *"we are interested in regions which exhibit slow intensity gradients"*, the schema selects various subsets of the evaluation functions which are stored in ILTM in order to determine whether or not the region token should be merged or resegmented. The results of these evaluation functions are combined, and stored in ISTM as hypotheses about the region tokens. The hypothesis values may then be accessed by other schemas of the system which will take the appropriate actions.

The schema for the **semantic region evaluation** utilizes sets of rules which evaluate the characteristics of regions, such as short line density, color, etc. to assign a plausible semantic label to a region (e.g. sky or tree). These rules, which are stored in the ILTM, are not as precise or complete as those used by a complete interpretation system ([2,30]), but provide valuable information for the intermediate-level schemas in the case that no overriding hypotheses have been specified as goal constraints.

The **region merge** schema is used by the system to reduce potential overfragmentation. By using goal constraints to select a subset of the evaluation functions stored in ILTM, instances of this schema are able to evaluate the desirability of a merge of a set of adjacent regions, and, if necessary, direct the actual merge process. The schema may either be applied in a general fashion over the entire image, evaluating all region tokens in the ISR for merge potential, or it may be invoked to directly merge a set of adjacent region tokens which some high-level process has deemed to be similar. Again using our example of a system with the *a priori* constraint to discriminate tree and sky, an instance of this schema would select two sets of merge criteria. In textured areas of the image, regions would be merged if they were adjacent, at least one of the tokens had a strong hypothesis for merge potential, and they both exhibited similar textural and hue characteristics. In non textured areas of the image, the merge criteria would still make use of adjacency and merge-hypothesis, but the regions would also have to demonstrate similar low values of intensity deviation as well as demonstrating co-planar intensity surface fits.

4.3 The line schemas

The **line segmentation** schema is designed to either directly insert lines into a region representation and thereby redefine the region mapping, or to control a region segmentation process which is designed to produce region boundaries which show a high degree of overlap with a set of specified lines. The former option is utilized in the case that there is strong belief in the existence of the lines, and when the specifications of the lines are known with a high degree of accuracy. Thus, given a set of lines which had been produced through the line extraction schema, and were therefore known to be present in the raw image data, the lines could be directly inserted into the region representation. A different rationale for this process would be a situation in which the position of a mobile robot was known with respect to a road, but the boundaries of the road were not immediately apparent in the robot's image data. In this case, the road boundaries could be determined from an internal map and then be placed directly into the segmentation data to aid in the interpretation of the remainder of the image.

This type of behavior by the schema was demonstrated in Figure 9, where the insertion of long lines restored several important boundaries which had been missed in the original region segmentation from Figure 7. Since the lines are used to split all regions with which they have significant intersection, several artificial boundaries have also were introduced by this process. However, as was shown in the final result (Figure 15), the region merging process was able to remove most boundaries which did not have a basis in the underlying data.

If a high-level interpretation schema concerned with shape were to make a hypothesis that a line might possibly exist in an area of the image, this type of direct insertion would not be desirable (i.e. the hypothesis might be incorrect). In this case, the characteristics of image feature distributions across the hypothesized line would be examined in an attempt to find a feature which could be used by a region segmentation process to produce a region mapping in which the line was matched by region boundaries.

The line tokens which are utilized by these schemas are either created as hypothesized lines by an interpretation schema or are extracted from the data by an instance of the **line extraction** schema, using one of several different straight line extraction algorithms ([4,35]). The schemas for **line-algorithm** and **line-feature** produce the necessary hypotheses for an instance of the line extraction schema, and these two schemas function in a fashion similar to their analogues for region segmentation. Figure 19 shows the set of lines produced by the line extraction schema, from which the set of long high contrast lines in Figure 8 were selected.

The low-level extraction processes controlled by instances of this schema may be constrained through goal constraints to restrict the output according to location, line orientation, line length, or contrast across the line. Thus a high-level interpretation process dealing with object shape which needed to find a particular line to complete a rectangle could post a goal to find a straight line in the image data which is similar to the desired line. On the other hand, a schema process which was interested in the short line density over an area of the image would invoke this schema with the constraint to find all possible lines. The set of line tokens would then be filtered on length, and perhaps contrast, to compute the density measure.

The **collinear line merge** schema is utilized in the case that a desired line has not been found intact by the line extrac-

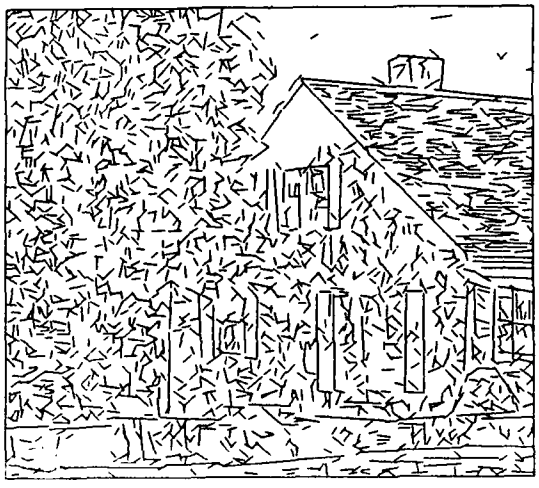


Figure 19: Lines Extracted by Line Extraction Schema

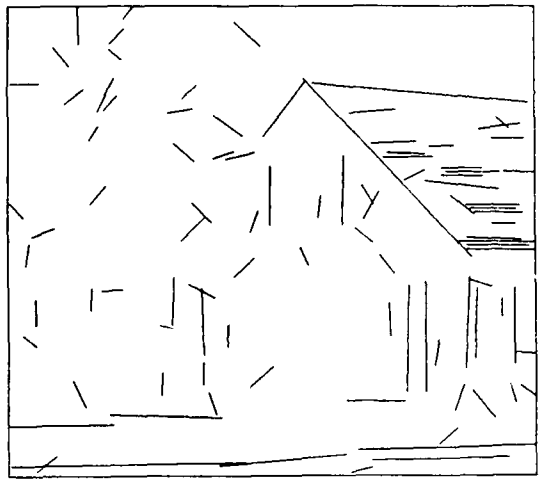


Figure 20: Collinear Line Grouping

tion processes. The low-level process controlled by this schema examines the set of existing line tokens which have been produced by the line extraction process to determine if suitable line tokens may be merged to produce the desired line ([35]). Figure 20 shows the long lines created by the application of the collinear line merge schema to the lines of Figure 8.

5. CONCLUSION

The set of schemas we have described is currently implemented within the GOLDIE system at the University of Massachusetts, and research is proceeding on the interface between this system and the high-level interpretation schemas being developed by other researchers in the VISIONS group. As currently implemented, the GOLDIE system provides an incomplete, yet powerful mechanism for the control of low-level image processing. Future development of the system to add schemas for additional low-level processing tasks, and to improve the computational efficiency, will produce a system which dramatically affects the overall image interpretation process.

There are four major aspects of GOLDIE which contribute to the utility of the system. Foremost is the fact that the system is goal-driven. This paradigm provides a coherent mechanism for top-down control in which low-level processing may be tuned to meet the expectations of higher level processes which are expressed through goal constraints. Second, the system makes use of a unified data representation which allows processes at any level of the interpretation hierarchy to create, access, or modify the data which represents the current state of interpretation processing. The third aspect is that the system contains an explicit representation of knowledge about low-level processes and the image domain which provides a flexibility permitting extension to additional processes or image domains. Finally, the GOLDIE system is capable of operating within a hypothesize-and-test protocol, exploring a variety of potential solutions to high-level requests rather than operating in a strictly deterministic manner.

REFERENCES

[1] Dana H. Ballard and Christopher M. Brown. *Computer Vision*. Prentice-Hall, Inc., Englewood Cliffs, New Jersey 07632, 1982.

[2] R. Belknap, E. Riseman, and A. Hanson. The information fusion problem and rule-based hypotheses applied to complex aggregations of image events. In *Proceedings of IEEE CVPR Conference*, pages 227–234, June 1986.

[3] Rodney A. Brooks and Thomas O. Binford. Interpretive vision and restriction graphs. In *Proceedings of The First Annual National Conference on Artificial Intelligence*, pages 21–27, August 1980.

[4] J. B. Burns, A. R. Hanson, and E. M. Riseman. Extracting straight lines. In *Proceedings Seventh International Conference on Pattern Recognition*, pages 482–485, Montreal, July 1984.

[5] John F. Canny. *Finding Edges and Lines in Images*. Technical Report 720, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, Massachusetts, June 1983.

[6] B. Draper, R. Collins, J. Brolio, J. Griffith, A. Hanson, and E. Riseman. Tools and experiments in knowledge based interpretation of road scenes. In *Proceedings: Image Understanding Workshop*, 1987. To be published.

[7] B. Draper, A. Hanson, and E. Riseman. *A Software Environment for High Level Vision*. Technical Report, Computer and Information Science Department, University of Massachusetts at Amherst, 1986. In preparation.

[8] Richard O. Duda and Peter E. Hart. *Pattern Classification and Scene Analysis*. John Wiley and Sons, New York, NY, 1973.

[9] Joey Griffith, Allen Hanson, and Edward Riseman. A rule based region merging system. In Preparation.

[10] Allen R. Hanson and Edward M. Riseman, editors. *Computer Vision Systems*. Academic Press Inc., New York, NY, 1978.

[11] Allen R. Hanson and Edward M. Riseman. Visions image understanding system - 1986. In Christopher M. Brown, editor, *Advances in Computer Vision*, Erlbaum Assoc., 1987. To be published.

[12] Allen R. Hanson, Edward M. Riseman, and Paul R. Nagin. *Region-growing in Textured Outdoor Scenes.* Technical Report 75C-3, Computer and Information Science University of Massachusetts at Amherst, Amherst Massachusetts 01003, 1982.

[13] Robert M. Haralick. Digital step edges from zero crossing of second directional derivatives. *IEEE Trans on Pattern Analysis and Machine Intelligence*, PAMI-6(1):58–68, January 1984.

[14] Robert M. Haralick and Linda G. Shapiro. Image segmentation techniques. *Computer Vision, Graphics, and Image Processing*, 29(1):100–132, January 1985.

[15] Vincent Shang-Shouq Hwang, Larry S. Davis, and Takashi Matsuyama. *Hypothesis Integration in Image Understanding Systems.* Technical Report TR-1137, Center for Automation Research, University of Maryland, College Park, Maryland 20742, June 1985.

[16] Charles A. Kohl. *Goal Directed Image Segmentation.* PhD thesis, Computer and Information Science Department, University of Massachusetts, Amherst, Massachusetts 01003, 1987. In Preparation.

[17] R. R. Kohler and A. R. Hanson. The visions image operating system. In *Proceedings Sixth International Conference on Pattern Recognition*, pages 71–74, Munich, October 1982.

[18] Ralf Kohler. A segmentation system based on thresholding. *Computer Graphics and Image Processing*, (15):319–338, 1981.

[19] Martin D. Levine and Samir I. Shaheen. A modular computer vision system for picture segmentation and interpretation. *Pattern Analysis and Machine Intelligence*, PAMI-3(5):540–554, September 1981.

[20] David Marr. *Vision.* W. H. Freeman, San Francisco, CA, 1982.

[21] M. Nagao and T. Matsuyama. *A Structural Analysis of Complex Aerial Photographs.* Plenum, New York, NY, 1980.

[22] Paul A. Nagin. *Studies in Image Segmentation Algorithms Based on Histogram Clustering and Relaxation.* Technical Report 79-15, Computer and Information Science University of Massachusetts at Amherst, Amherst Massachusetts 01003, 1979.

[23] Ahmed M. Nasif. *A Rule-Based Expert System for Image Segmentation.* PhD thesis, Electrical Engineering Department, McGill University, Montreal, Canada, March 1983.

[24] Ronald B. Ohlander. *Analysis of Natural Scenes.* PhD thesis, Department of Computer Science, Carnegie-Mellon University, Pittsburgh, PA 15213, April 1975.

[25] Yu-ichi Ohta. *Knowledge-based Interpretation of Outdoor Natural Color Scenes.* Pitman Advanced Publishing Program, Boston, MA, 1985.

[26] Yu-ichi Ohta. *A Region-Oriented Image-Analysis System by Computer.* PhD thesis, Kyoto University Department of Computer Science, 1980.

[27] Cesare C. Parma, Allen R. Hanson, and Edward M. Riseman. *Experiments in Schema-Driven Interpretation of A Natural Scene.* Technical Report 80-10, Computer and Information Science Department - University of Massachusetts at Amherst, Amherst, Massachusetts 01003, April 198.

[28] George Reynolds and J. Ross Beveridge. Searching for geometric structure in images of natural scenes. In *Proceedings: Image Understanding Workshop*, 1987. To be published.

[29] George Reynolds, Nancy Irwin, Allen Hanson, and Edward Riseman. Hierarchical knowledge-directed object extraction using a combined region and line representation. In *IEEE 1984 Proceedings Of The Workshop On Computer Vision Representation And Control*, pages 238–247, IEEE, 1984.

[30] Edward M. Riseman and Allen R. Hanson. *A Methodology for the Development of General Knowledge-Based Vision Systems.* Technical Report 86-27, Computer and Information Science University of Massachusetts at Amherst, Amherst Massachusetts 01003, 1986.

[31] A. Rosenfeld and A. Kak. *Digital Picture Processing.* Academic Press, New York, NY, 1976.

[32] Bert Shaw. *Some Remarks on the Use of Color for Machine Vision.* Technical Report 83-31, Computer and Information Science Department - University of Massachusetts at Amherst, Amherst, Massachusetts 01003, September 1983.

[33] J. M. Tenenbaum and H. G. Barrow. *Experiments in Interpretation-Guided Segmentation.* Technical Report 123, SRI International, 333 Ravenswood Ave., Melno Park, CA 94025, March 1976.

[34] William B. Thompson and Albert Yonas. What should be computed in low level vision. In *Proceedings of The First Annual National Conference on Artificial Intelligence*, pages 7–10, August 1980.

[35] Richard Weiss and Michael Boldt. Geometric grouping applied to straight lines. In *Proceedings of IEEE-CVPR Conference*, pages 489–495, June 1986.

[36] Terry E. Weymouth. *Using Object Descriptions in a Schema Network for Machine Vision.* PhD thesis, University of Massachusetts, Amherst, Massachusetts 01003, 1986.

[37] Y. Yakimovsky. *Scene Analysis Using a Semantic Base for Region Growing.* PhD thesis, Stanford University, Palo Alto, CA, 1973.

ACTIVE VISION

John (Yiannis) Aloimonos
Isaac Weiss
Center for Automation Research
University of Maryland
College Park, MD 20742

Amit Bandyopadhyay
Department of Computer Science
SUNY Stony Brook
Stony Brook, NY 11790

ABSTRACT

We investigate several basic problems in vision under the assumption that the observer is active. An observer is called active when engaged in some kind of activity whose purpose is to control the geometric parameters of the sensory apparatus. The purpose of the activity is to manipulate the constraints underlying the observed phenomena in order to improve the quality of the perceptual results. For example a monocular observer that moves with a known or unknown motion or a binocular observer that can rotate his eyes and track environmental objects are just two examples of an observer that we call active. We prove that an active observer can solve basic vision problems in a much more efficient way than a passive one. Problems that are ill-posed, nonlinear or unstable for a passive observer become well-posed, linear or stable for an active observer. In particular, the problems of shape from shading and depth computation, shape from contour, shape from texture and structure from motion are shown to be much easier for an active observer than for a passive one. It has to be emphasized that correspondence is not used in our approach, i.e., active vision is not correspondence of features from multiple viewpoints. Finally, active vision here does not mean active sensing.

1. INTRODUCTION AND MOTIVATION

Most past and present research in machine perception has involved analysis of passively sampled data (images). Human perception, however, is not passive, it is active. Perceptual activity is exploratory and searching. When humans see and understand, they actively look. In the process of looking, their eyes adjust to the level of illumination, focus on certain things, converge or diverge and their heads move to obtain a better view of the scene.

It is very natural to ask why human observers operate in such a way, because certainly humans are very efficient in visual tasks. In other words, how does the fact that an observer is active affect the levels of a visual system (as described by Marr, 1982), namely computational theory, representations and processing algorithms and implementation. Does an active observer have any advantage over a passive observer, in any computational theoretic, algorithmic or implementational way? In this research, we examine this question and we find that indeed an active observer has a great deal of advantage as far as the first two levels of the visual system are concerned (computational theory, algorithms). A natural way to examine this question is to study basic problems of vision whose solutions demonstrate visual abilities, avoiding in this way the potential philosophical snare of getting into a discussion of the vision problem in general.

Another motivation for examining active vision is the fact that passive vision has been shown to be very problematic. Almost every basic problem in passive machine perception is very difficult, because it is ill-posed in the sense of Hadamard (1923). So, because there does not exist a unique solution, the problem has to be regularized, by imposing additional constraints, which should be physically plausible. An example of such an additional constraint is some kind of smoothness of the unknown functions. There has been excellent research in regularizing early vision problems, originated in (Poggio et al., 1986; Poggio and Koch, 1985). Even though the regularization paradigm is very attractive for its mathematical elegance and for being a legitimization of already published research (Horn and Schunck, 1981; Ikeuchi and Horn, 1981; Hildreth, 1984), it has some shortcomings, in the sense that it cannot deal with the full complexity of vision. One problem is the degree of smoothness required for the unknown function that has to be recovered; for example, some unrealistic results have been reported in surface interpolation, because depth discontinuities are smoothed too much. Research on regularization in the presence of discontinuities, while pioneering, is still premature (Terzopoulos, 1984; Lee and Pavlidis, 1986). Another problem is that standard regularization theory deals with linear problems and is based on quadratic stabilizers. In the case of nonquadratic functionals standard regularization theory may be used, but the situation is problematic (Morozov, 1984). For non-quadratic functionals, the search space may have many local minima and in this case only stochastic algorithms might have some success (Kirkpatrick et al., 1984). Aside from the fact that most passive vision problems are ill-posed, some well posed problems are very unstable. That is to say, even if from the physical constraints the problem is

shown to have a unique solution, finding this solution is very difficult and unstable, in the sense that a small error in the input of the perceptual process can create catastrophic results in the output. An example of such a problem is the passive navigation problem (Ullman, 1977; Tsai and Huang, 1984; Bruss and Horn, 1984; Waxman and Ullman, 1985; Longuet-Higgins, 1981; Longuet-Higgins and Prazdny, 1982), where retinal motion (retinal velocities or displacements) are used as the input. In contrast, it will be shown that some problems that are ill-posed for a passive observer become well-posed for an active one, and problems that are unstable become stable.

We now set out to examine the advantages of an active observer with respect to the *computational theory* and *algorithms* levels of visual systems. In particular we show that:

(1) The problem of shape from shading in the case of a passive observer has infinitely many solutions and additional assumptions are required to guarantee uniqueness. Furthermore, the stability of the developed algorithms is in question. In contrast, in the case of an active observer, the shape from shading problem is shown to have a unique solution. In particular the commonly used assumptions of smoothness of the visible surface and constant albedo become unnecessary. This makes it possible to deal with complex shapes having discontinuities of surface orientation and reflectance. The isotropic radiance constraint can be relaxed to deal with partly specular surfaces. Our method is not susceptible or prone to instabilities. Depth computation is addressed also.

(2) The problem of shape from contour is a difficult one for a passive observer since assumptions have to be employed to obtain a unique solution. We show that in the case of an active observer the problem has a unique solution, which, moreover, can be found using linear equations. The stability of the proposed computation is also examined.

(3) The problem of shape from texture requires assumptions about the texture to make it solvable by a passive observer. We prove that an active observer can recover shape from texture without any assumptions and using linear equations.

(4) Finally, the problem of structure from motion has been shown to be very unstable. We show that an active observer can recover structure from motion by using linear equations.

The following table compares the performance of a passive and an active observer in the solution of several basic problems.

In the following sections we study the basic problems described above. The case of the computation of optic flow will be described in subsequent publications. In all cases, we try to avoid solving the correspondence problem. Also, at this point it should be clear that active vision is not active sensing; it is just vision in which physical constraints are simplified because the observer can change state in an active way.

2. SHAPE FROM SHADING

2.1. Introduction

In the problem of recovering shape from shading, the input consists of the brightness at each point of an image, or images, and the desired output is the depth and/or the surface normal of the corresponding point on the visible surface. In principle, the depth map (the depth z as a function of the x,y coordinates) contains all the information about the surface and the surface normals can be computed directly from the knowledge of the depth map. In practice, however, those depths cannot be derived with sufficient accuracy for calculating the normals and one would like to infer the normals directly from the image. In the following we briefly discuss the current approaches to solving the problem and their severe limitations, and then show how the active vision paradigm overcomes these limitations.

The simplest approach to the shape from shading problem involves using one image of the surface. To find a solution, the following assumptions are commonly used, none of which is particularly valid in a realistic situation:

1) The surface is smooth
2) The surface reflectance characteristics, usually Lambertian, are the same throughout the surface.
3) The lighting, usually one point light source, is the same throughout the surface.
4) The image is nearly noise-free.

Based on these assumptions, one can write a functional of the surface and its normals, which should be minimal for the correct surface. The functional is in general a non-linear function of a large number of unknowns (depth and normals at each point), so it is very hard to achieve convergence of the numerical optimization to the global minimum. Very good research along this line is described in Horn (1977) and Ikeuchi and Horn (1981). But in this case, noise compounds the problem, creating instabilities. So these techniques have had only very limited success.

An improvement can be achieved by using two images of the surface. Combining information from the two images makes it easier to solve the minimization problem. In this case one does not need to consider the whole image at once during the minimization, as in the previous case, because the image can be decoupled into narrow strips along epipolar lines. Only points along a pair of such lines need be considered at the same time. However, to be able to take advantage of the two views, one must first find the correct correspondence between the two images. One can distinguish between two methods in using more than one image:

a) The two cameras are very close to each other. In this case it is easy to establish a correspondence between points in the two images. Moreover, the equation one needs to solve are linear, because the small distance allows use of first order Taylor expansion of the various functions involved. However, the small baseline

Table 1

Problem	Passive Observer	Active Observer
Shape from shading	Ill-posed problem. Needs to be regularized. Even then, unique solution is not guaranteed because of non-linearity.	Well-posed problem. Unique solution. Linear equation used. Stability.
Shape from contour	Ill-posed problem. Has not been regularized up to now in the Tichonov sense. Solvable under restrictive assumptions.	Well-posed problem. Unique solution for both monocular or binocular observer.
Shape from texture	Ill-posed problem. Needs some assumption about the texture.	Well posed problem. No assumption required.
Structure from motion	Well posed but unstable. Nonlinear constraints.	Well posed and stable. Quadratic constraints, simple solution methods, stability.
Optic flow (area based)	Ill-posed. Needs to be regularized. The introduced smoothness might produce erroneous results.	Well posed problem. Unique solution. Might be unstable.

between the cameras severely limits the accuracy of the method.

b) The cameras are far apart. This leads to more accurate results, provided one can solve the correspondence problem. This problem has proven to be very difficult, and the techniques that deal with it are far from satisfactory (for details see, e.g. Horn (1986).

The AV (Active Vision) paradigm has the advantages of both methods, without their shortcomings. First, having multiple viewpoints can solve the correspondence problem. In fact, it can be shown that three cameras are enough to resolve most of the correspondence ambiguities in the Lambertian case. The stability and reliability also increase. But multiple viewpoints are not enough by themselves to make a method work. This is because we again run up against the problem of non-linear optimization, which, with so many variables, rarely converges to the global minimum without a very good initial guess. The key to the success of AV is the fusion of the long- and short-baseline methods. We have available images taken at short intervals, as well as images separated by long intervals, and we can use information from both.

The AV method can thus proceed in two stages:

1) A short baseline stage, in which a succession of frames taken at short distances apart is examined. This stage, being linear, is easily solvable, thereby providing initial estimates for the depths and surface normals.

2) A long baseline stage. Now that an estimate exists, it can be used in several ways, as we shall see. In the Lambertian case we use it to establish correspondence between points seen from far-away viewpoints, while in the non-Lambertian case it is the initial guess in a non-linear optimization procedure.

We shall show that with this method we can recover the geometry of the visible object at each individual point independently. We do not need the assumption mentioned before, of global optimization (maximum smoothness) of the whole object. Moreover, it turns out that in spite of the greater amount of data, our task is much easier than in the previous methods, since the recovery process can be done for each point separately, rather then having to deal with the image as a whole. All this is done in a stable and noise-resistant way.

2.2. Geometrical Preliminaries

We work in perspective projection. For simplicity we assume that the camera moves with its optical axis remaining parallel to the z axis, and the lens is in the x,y plane. (Rotations will not get in the way of the basic principles.) The coordinates of the camera's lens center, x_c, y_c, are moving with a known motion, causing changes in the brightness of the image points. The coordinates of the image points are measured in the fixed coordinate system, regardless of the camera's position, and are denoted by x_i, y_i (with $z_i = -1$, i.e. a focal length of 1). The coordinates of a point on the real object are denoted by x_o, y_o, z_o. One has the relation between the object, camera and image coordinates:

$$x_i - x_c = -\frac{x_o - x_c}{z_o} \tag{2.1}$$

$$y_i - y_c = -\frac{y_o - y_c}{z_o}$$

A fixed camera forms a brightness function $E(x_i,y_i)$ in the image plane. This function changes when the camera moves (i.e. when x_c,y_c change), and the brightness now depends on four variables: $E(x_i,y_i;\, x_c,y_c)$. There are two possible ways to represent the change. (1) Measure the change at every point x_i,y_i of the image, i.e. find the partial derivatives of E with respect to x_i,y_i, and also with respect to x_c,y_c. This leads to optical flow-like methods. In the fluid mechanical analogy, this is a representation of the flow in the Euler coordinate system. (2) Follow a point with a given brightness along successive frames. This is analogous to following a particular element of the fluid along its path of motion and recording this element's coordinates and their derivatives. This is known as the Lagrangian system. The two methods are of course mathematically equivalent, and the choice between them depends on convenience in a particular situation. For a Lambertian surface, the Lagrangian system has an obvious advantage, because an object point projects into image points of equal brightness in all images. Thus, following a point of a given brightness along successive frames means following the same object point. It is also preferable in the non-Lambertian case, in a modified form, as we shall see.

When we have two viewpoints, with either a short or long baseline, two matching systems of epipolar lines are created, one for each image. Unlike the single image method, in which the image has to be treated as a whole, the two camera method makes it possible to decouple the problem into reconstruction of shape along epipolar lines. In more detail, consider a plane containing the two lens center points of the cameras. It intersects the two image planes in straight lines, namely two matching epipolar lines. A point on the epipolar line in one image corresponds to a point somewhere on the matching epipolar line on the other image (belonging to the same plane); thus we only have to find the correspondence of points along epipolar lines, which can offer a significant simplification.

In the following, we treat the Lambertian and non-Lambertian cases differently. The isotropy of the Lambertian reflectance function presents us with both a simplification and a difficulty. The simplification was mentioned above, namely the constant brightness in all images of a particular object point. The difficulty is that the parameters that influence the brightness, such as the surface orientation and the light direction, cannot be recovered by changing the point of view, as the reflectance function is independent of the point of view. To recover the orientation, we are forced to use spatial derivatives of the brightness. Happily, this makes the problem linear even for the long baseline step (at least in our particular geometry).

We will now make the above arguments more formal. The image brightness is governed by the "image irradiance equation" which is easily generalized to our case:

$$E(x_i,y_i;x_c,y_c) = R(\hat{n},\hat{v},\vec{s}) \qquad (2.2)$$

The left hand side is measured from the image at each camera location x_c,y_c. The right hand side contains our assumptions about the light reflectance properties of the surface. This reflectance depends, in general, on the surface orientation $\hat{n}$, on the viewing direction $\hat{v}$, and on a vector containing a finite number of parameters, $\vec{s}$, which represents the light distribution and the surface intrinsic properties. The variables in R were separated in this way because $\hat{n}$ is the unknown that we are mainly interested in, and $\hat{v}$ is a parameter under our control, as the camera moves and changes the viewing direction. ($\hat{v}$ is the direction of the known vector $(x_i,y_i)-(x_c,y_c)$.) In view of the above, recovering shape from shading amounts to solving this image irradiance equation (for $\hat{n}$).

If the correspondence problem were solved, we could use the above equation for the same object point in different views, i.e. different $\hat{v}$s. We thus obtain a set of equations for the unknowns in R, in particular $\hat{n}$. Whether this set equations is degenerate depends on the particular R. For a typical R, $\hat{n}$ and $\hat{v}$ are coupled in a term of the form $\hat{n}\cdot\hat{v}$, so the equations are not degenerate, at least with respect to finding $\hat{n}$. For a Lambertian surface, however, the reflectance is independent of $\hat{v}$, and $\hat{n}$ cannot be found in this way.

2.3. Lambertian Surfaces

The Lambertian case is special in that the shape reconstruction process can be decoupled not only along epipolar lines but also along isophotes. That is, when a contour of equal brightness moves in the image (with the movement of the camera), the new contour corresponds to the same set of object points as the old contour. This is because a Lambertian surface element is seen with the same brightness from every point of view. Thus, it is convenient to parametrize the image with a set of coordinates consisting of the epipolar lines and the isophotes. One can assign some labeling to the isophotes, which we denote by α, and a labeling to the epipolar lines, denoted by β. The particular labeling mapping is immaterial, as long as it is well behaved and increases monotonically (with respect to some spatial ordering of these coordinate lines). Such a well behaved mapping is always possible at least at some neighborhood, which is what we need for taking derivatives. The coordinate lines (epipolar lines and isophotes) move and change their shape as the camera moves, but they keep their labels α,β. This is in accordance with the Lagrangian coordinate representation. The key advantage of the scheme is that an image point labeled α,β always corresponds to the same object point. The correspondence problem is then essentially solved. We can now attach the labels (α,β) to the object point (x_o,y_o,z_o) also. (Two views are needed to create epipolar lines. We can take one view as fixed, say at the beginning of the movement, while the other moves. We will not need the image from the fixed view. It only serves to create consistent systems of epipolar lines.)

By measuring the position in the image of the point labeled α,β as the camera moves, i.e. the functions $x_i(\alpha,\beta,x_c,y_c)$, $y_i(\alpha,\beta,x_c,y_c)$ and their derivatives, one can infer the depth and normal at the corresponding object point. As we shall see, the derivatives with respect to (x_c,y_c) are not needed except in find an initial guess, but the derivatives with respect to α,β are the ones that enable us to recover the normal. Differentiating relation (2.1) we obtain

$$\begin{aligned}\frac{\partial x_i}{\partial \alpha} &= -\frac{1}{z_o}\frac{\partial x_o}{\partial \alpha} + \frac{x_o - x_c}{z_o^2}\frac{\partial z_o}{\partial \alpha} \\ \frac{\partial y_i}{\partial \alpha} &= -\frac{1}{z_o}\frac{\partial y_o}{\partial \alpha} + \frac{y_o - y_c}{z_o^2}\frac{\partial z_o}{\partial \alpha}\end{aligned} \tag{2.3}$$

$$\begin{aligned}\frac{\partial x_i}{\partial \beta} &= -\frac{1}{z_o}\frac{\partial x_o}{\partial \beta} + \frac{x_o - x_c}{z_o^2}\frac{\partial z_o}{\partial \beta} \\ \frac{\partial y_i}{\partial \beta} &= -\frac{1}{z_o}\frac{\partial y_o}{\partial \beta} + \frac{y_o - y_c}{z_o^2}\frac{\partial z_o}{\partial \beta}\end{aligned} \tag{2.4}$$

where the left-hand side quantities are measured from the image, and the right-hand side contains the unknowns.

Defining the two dimensional vectors $\vec{x}_o = (x_o,y_o)$, $\vec{x}_i = (x_i,y_i)$, and $\vec{x}_c = (x_c,y_c)$, we substitute eqs. (2.1) in eqs. (2.3,4) to eliminate the $1/z_o^2$ factor and obtain

$$\frac{\partial \vec{x}_i}{\partial \alpha} = -\frac{1}{z_o}\frac{\partial \vec{x}_o}{\partial \alpha} - \frac{\vec{x}_i - \vec{x}_c}{z_o}\frac{\partial z_o}{\partial \alpha} \tag{2.5}$$

and a similar equation in β. Using this equation is the key idea in recovering the surface orientation in the Lambertian case. Its intuitive interpretation is that as the camera moves, the infinitesimal distances $\partial\alpha,\partial\beta$ between two object points do not change, but the corresponding $\partial\vec{x}_i$, i.e. the (geometrical) distances between the corresponding isophotes (or epipolars) in the image, do change. This change depends on the geometry of the object, as is seen in the above equation, and thus this geometry can be recovered.

For a short baseline, we calculate the change of the above derivatives caused by the camera's movement by differentiating with respect to $\vec{x}_c$:

$$\frac{\partial^2 \vec{x}_i}{\partial \alpha \partial \vec{x}_c} = -\left(\frac{\partial \vec{x}_i}{\partial \vec{x}_c} - \delta_{ij}\right)\frac{1}{z_o}\frac{\partial z_o}{\partial \alpha} \tag{2.6}$$

where δ_{ij} is the Kronecker delta, and the indices i,j can take either of the two values x or y. We can multiply eqns. (2.1),(2.5),(2.6) by z_o to obtain the linear system of equation:

$$(\vec{x}_i - \vec{x}_c)z_o + \vec{x}_o - \vec{x}_c = 0 \tag{2.7}$$

$$\frac{\partial \vec{x}_i}{\partial \alpha}z_o + \frac{\partial \vec{x}_o}{\partial \alpha} + (\vec{x}_i - \vec{x}_c)\frac{\partial z_o}{\partial \alpha} = 0 \tag{2.8}$$

$$\frac{\partial^2 \vec{x}_i}{\partial \alpha \partial \vec{x}_c}z_o + \left(\frac{\partial \vec{x}_i}{\partial \vec{x}_c} - \delta_{ij}\right)\frac{\partial z_o}{\partial \alpha} = 0 \tag{2.9}$$

The solution can now proceed in two steps:

1) Solve the linear system of six equations (2.7,8,9) for the six unknowns $\vec{x}_o$, z_o, $\partial\vec{x}_o/\partial\alpha$, $\partial z_o/\partial\alpha$. This is an inhomogeneous system of rank six (in general) and thus has a unique solution. The coefficients of the unknowns do not have to come from measurements at one point. (By measurements we mean the functions $\vec{x}_i(\alpha,\beta)$ and their derivatives.) Rather, measurements can be taken from several points along the path of the point $\vec{x}_i(\alpha,\beta)$ and averaged. As the equations are linear, the averaged measurements can be substituted in it so that the system need be inverted only once. Thus we have obtained a good estimate of the unknowns.

2) The accuracy of the previous step may not be good, as we used a second derivative (eq. (2.9)). This amounts to using a short baseline stereo technique. For better accuracy we can use the first four equations, (2.7) and (2.8), at two far away points (or camera locations) $\vec{x}_c$ and $\vec{x}_c'$. This will result in eight equations, two of which are superfluous. This step needs the establishment of a correspondence between the two views. Since we already know the location of the object points roughly from the first step, applied to both images with $\vec{x}_c$ and $\vec{x}_c'$, any correspondence ambiguities are resolved. Without the first step, correspondence ambiguities could arise from having several points with the same brightness along one epipolar. (Alternatively to the first step, we could simply trace the point (α,β) from frame to frame, but then we may need continuous, uninterrupted tracing.) Now the longer baseline will significantly improve the accuracy and stability. Equation (2.9), involving the derivatives with respect to $\vec{x}_c$, is no longer part of the system. Since we still have a linear system of equations, we can again make use of averaged measurements with different base points, and invert a system of equations with the coefficients calculated from the averaged measurements.

The above steps will yield the location of the object point x_o,y_o,z_o and one tangent on the surface, namely $\frac{\partial x_o}{\partial \alpha},\frac{\partial y_o}{\partial \alpha},\frac{\partial z_o}{\partial \alpha}$. A similar procedure using the β parameter will give another tangent. Once the tangents of the surface at $\vec{x}_o,z_o$ are known, the normal can immediately be calculated by their vector product

$$\hat{n} = A^{-1/2}\left(\frac{\partial x_o}{\partial \alpha},\frac{\partial y_o}{\partial \alpha},\frac{\partial z_o}{\partial \alpha}\right) \times \left(\frac{\partial x_o}{\partial \beta},\frac{\partial y_o}{\partial \beta},\frac{\partial z_o}{\partial \beta}\right)$$

where A is the scalar product of the above tangent vectors.

A few points are worth noting:

1) At no point did we need to know either the albedo (the surface reflectance) or the light characteristics. They can both vary from point to point on the surface without affecting the calculation. Thus, when a change in the brightness is detected, we are able to

tell whether it results from a change in the geometry of the surface or from the reflectance or lighting (and can calculate the local geometry). This is unlike other theories.

2) The surface is not necessarily smooth, as is assumed in most shape reconstruction theories. If one of the derivatives used is too large, we simply label this point as a discontinuity and apply our method to other points in the neighborhood.

3) The tangents have not been derived from the depth map, which would have been quite inaccurate. They were independent variables in a system of equations that determined both the depth and the tangents. In would be of interest to carry out an error analysis of the results.

4) The equations for both the short and long baseline turned out to be linear, which frees us from the recurring hard problems of non-linear optimization.

5) Using the brightness function at several viewpoints was not enough to recover the surface normal, and we needed its derivatives $\partial\vec{x}_i/\partial\alpha, \partial\vec{x}_i/\partial\beta$. The infinitesimals $\partial\alpha, \partial\beta$ do not change from one image to another, but $\partial\vec{x}_i$ changes in a way dependent on the normal, which can thus be recovered. We will not need the derivatives in the non-Lambertian case (except at the first stage).

6) The formalism is applicable to any contours, not necessarily isophotes. Many contours that are marked on the object can be detected on an image by means other than changes in shading. For instance, contours can be formed by changes in texture or color. If the change is continuous, such as a gradual color change, we can draw contours on which some property such as color remains constant. We can then label the contours in the same way we labeled the isophotes and apply the above formalism without change. We can thus find the position and orientation of the visible surface elements. If the change that produced a contour was a sharp discontinuity, the contour is isolated. In this case we cannot measure the derivative of the contour's property along epipolar lines, the way we measured the derivative of the isophote's brightness. We can still measure the other derivative, i.e. of the epipolar line along the contour. (Formally, we can differentiate with respect to β, but not α). This is enough to find the position and direction of the contour element (by solving eqs. 2.7,8,9 with β). This line element lies on a visible surface element, and its direction is the direction of one tangent to that surface. In this case, then, without further information, the surface element's orientation can be determined only up to rotation around the contour element.

2.4. Non-Lambertian Surfaces

When the reflectance function has a non-isotropic component, the reconstruction becomes more difficult. The isophotes at different viewpoint no longer correspond to the same object point. Thus the simple Lagrangian formalism described above has to be modified. Additionally, more unknowns are added to the problem, as there are more parameters in the reflectance function, including the relative strengths of the nonisotropic components. (They enter the vector $\vec{s}$ in eq. (2.2).) This reflectance function is in general non-linear. We think that the Lagrange formalism, namely following points on the changing image, is still preferable to the Euler formalism used in most optical flow theories (measuring changes at a fixed point on the image) because it enables us to deal with an object point (and its neighborhood) separately from other points, while a fixed point on the image corresponds to different object points during the motion of the camera.

Because simply following the isophotes is not useful as in the Lambertian case, the image brightness E has to be taken into account explicitly. It is useful to work in a five-dimensional vector space V, whose components are $E, \vec{x}_i, \vec{x}_c$. In the fluid mechanical analogy, the subspace spanned by first three components, $E, \vec{x}_i$, is somewhat analogous to a "phase space", while the $\vec{x}_c$ represents temporal dimensions. Thus, for one viewpoint (fixed $\vec{x}_c$), the image is a 2-D surface in the above 3-D subspace. It is simply the surface described by brightness function $E(x_i, y_i)$. As the camera moves along some (known) path $\vec{x}_c(\gamma)$ in the $\vec{x}_c$ dimensions, it creates a one-parameter family of such 2-D surface, so that we have a 3-D surface in the 5-D space. This surface is our data, measured from the images. We shall call it the "brightness surface", to distinguish it from the visible surface.

Consider again the image irradiance equation

$$E(x_i, y_i; x_c, y_c) = R(\hat{n}, \hat{v}, \vec{s}) \qquad (2.2)$$

where the viewing direction $\hat{v}$ depends on $\vec{x}_i, \vec{x}_c$:

$$\hat{v} = \frac{\vec{x}_i - \vec{x}_c}{|\vec{x}_i - \vec{x}_c|}$$

This equation has to be solved for each object point. We assume that the functional form of R is the same at each point, with different parameters. The 3-D brightness surface appears on the left-hand side of this equation. To solve the equation, we need to represent the right hand side too. This can be done by representing each visible surface element (around an object point) as a trajectory in the 5-D space. For such an element, all the unknowns $\hat{n}, \vec{s}, \vec{x}_0, z_0$ are fixed. The camera moves along the path $\vec{x}_c(\gamma)$, causing a simultaneous move $\vec{x}_i(\gamma)$ in the image coordinates, in accordance with the perspective geometry equation, eq.(2.1). The light reflected by the element in the viewing direction also changes. Using the reflectance function R, we can compute $R(\gamma)$. Now we can plot a trajectory in the 5-D space, with R being plotted in the E dimension. We obtain the curve

$$\Gamma(\gamma) = (\vec{x}_c(\gamma), \vec{x}_i(\gamma), R(\gamma))$$

In summary, we have a trajectory $\Gamma(\gamma)$ for every set of unknown parameters pertaining to one visible surface element.

If such a surface element lies on our visible object, then its trajectory in the 5-D space V must lie on the 3-D brightness surface (representing E in that space). This is is because of the equality $E = R$, i.e. the reflectance calculated for the surface element has to match the measured image brightness, in every image observed.

Since the visible object is made of such elements, the 3-D brightness surface created by the object in the 5-D space is made up of such individual trajectories. We can distinguish between the different trajectories by their starting point. Looking at the first image on the camera's path, each point $\vec{x}_i$ belongs to one surface element, and can be used to label the corresponding trajectory.

The solution of the image irradiance equation $E = R$ for each surface element is now reduced to finding a set of unknowns $\hat{n}, \vec{x}_o, z_o, \vec{s}$ for which the element's trajectory in the 5-D space lies entirely on the brightness surface (and has a given starting point). Correspondence is not an issue here. A trajectory in V immediately defines a correspondence between successive images. Stated differently, the correspondence problem has been merged into the trajectory matching problem.

So, by following trajectories that are generated by single object points, we have been able to decouple the reconstruction problem and solve separately for small sets of parameters, belonging to each object point, rather then dealing with the image as a whole. The usual theories, in contrast, lead to a large set of coupled equations involving all the image points. This trajectory following method can be associated, in a way, with the Lagrange system of fluid mechanics.

Having clarified the theoretical vision aspect of the problem, we have now to deal with the more practical problem of fitting a trajectory to a surface. First, how do we define fitting? One way is to demand that the equation $E = R$ be satisfied at several points along the trajectory. Thus we obtain a set of equations, one from each such point, for the set of unknowns. If the number of points is at least as large as the number of unknowns, a solution can be found, in a generic case. If it is larger, the reliability improves. This is similar to the situation described in Section 2. As noted there, the system is not degenerate because the geometrical unknowns, $(\hat{n}, \vec{x}_o, z_o)$, are coupled to the viewing direction $\hat{v}$. The other parameters in R which have some coupling to the viewing direction will also be recovered. For instance, having a single light source, we may find its direction, and find the product of the intensity with the surface albedo, but the latter cannot be separated in this method.

In practice, because of noise and other inaccuracies, it may be impossible to find such a solution. Thus it is preferable to turn the problem into one of optimization. Unlike theories such as regularization, this optimization is not a result of some additional assumptions such as smoothness, which are not needed here. It is simply a way to make the tolerance limit of the fitting less stringent.

One functional we can optimize is the sum of distances from each trajectory point to the nearest brightness surface point. For simplicity, we measure the distance in the E dimension only, taking the coordinates $\vec{x}_c, \vec{x}_i$ to be the same for both the trajectory and the surface. Thus, we seek to minimize the functional

$$\int_{\Gamma(\gamma)} \left(R\left[\vec{x}_i(\vec{x}_o, z_o), \vec{x}_c, \hat{n}, \vec{s}\right] - E(\vec{x}_i, \vec{x}_c) \right)^2 d\gamma$$

over all possible values of the unknowns $\vec{x}_o, z_o, \hat{n}, \vec{s}$, with the constraint of passing through a given starting point. The unknowns enter the problem through both their explicit appearance in the integrand and their determination of the trajectory $\Gamma(\gamma)$. (Recall that $\vec{x}_i$ is determined by eq. (2.1), which contains the unknown position $\vec{x}_o, z_o$ of the object point).

Although the number of unknowns is small, we may still face difficulties resulting from the non-linear nature of the problem. Our strategy to deal with that is similar to the one we used in the Lambertian case. In step 1, we use a very short trajectory, and linearize our expressions, using a Taylor expansion of R. Alternatively, higher derivatives can used instead of several close points along the trajectory. The solution of the linear equations provides a good initial guess for step 2.

Step 2 is the non-linear optimization described above. The small number of unknowns and the good initial guess make the task quite easy.

As in the Lambertian case, no use has been made of a smoothness assumption. Furthermore, the position in space, as well as the orientation, have been recovered for each object point separately. The other parameters in the reflectance function which are coupled to the viewing direction are also recovered. The multiple viewpoints make the result reliable and stable. For the Lambertian case this method has a certain degeneracy, as $\hat{n}$ and $\vec{s}$ cannot be separated solely by moving the camera, as noted before.

2.5. Summary and Conclusions

In evaluating the AV paradigm for recovering shape from shading, two major benefits arise:

1) More viewpoints give us more information, and allow us to dispose of the restrictive assumptions used in previous research and increase the stability with respect to noise and other errors.
2) One would think that with more images the task of processing the given information will be harder then for one or two viewpoints. In fact, just the opposite has happened, because we have been able to decouple the handling of the individual object points, and solve the problem in small, separate parts.

These advantages together allow us to infer the geometrical parameters as well as reflectance function parameters at each point individually. This means that we recover the true shape of the object, rather then a smoothed and "optimized" version of it, as other theories do. This also resolves the perennial problem of whether a

change in the image brightness is caused by a change in the object's geometry, or by other light reflectance factors.

3. SHAPE FROM CONTOUR

Here we study the problem of the detection of shape from contour by an active observer, and we compare the performance of this new active scheme with the passive approach. We consider the contour to be planar. Work on nonplanar contours can be found in (Ito and Aloimonos, 1987a).

3.1. Introduction

The human perceiver is able to derive enormous amounts of information from the contours in a scene. As part of this capacity, we are able to use the shapes of image contours (as they are seen by both eyes) to infer the shapes and dispositions in space of the surfaces they lie on, as well as their motion. The interpretation of contours by a binocular observer involves several subproblems (following Witkin, 1981):

a) *Locating contours in the images*

If contours are to be used to infer anything, they must be found. The human perceiver has little difficulty deciding what is and is not a contour, yet the automatic detection of edges has proved very difficult. Perhaps this fact should not be surprising; the contours that we see in natural images usually correspond to definite physical events, such as shadows, depth discontinuities, color differences and the like. Our ability to detect these events may say more about their significance for image interpretation than about their ease of detection. Why should we expect events that have simple descriptions in terms of the structure of the scene to have simple descriptions in terms of the image intensity as well? If the physical significance of contours is taken as their primary feature, then at least we know what is being detected, even if we don't know how. But recent research (Nalwa, 1985) shows that we are reasonably well advanced as far as detection of contours goes. Actually, we can say that we can fairly well detect the contours in an image, even if there are some inaccuracies.

b) *Labeling contours (i.e. distinguishing contours which are due to different physical events)*

If contours correspond to different physical events, then an essential component of their interpretation must be to decide which contours denote which event, since each kind of contour imparts a different meaning. Recent work has shown that strong structural constraints can be applied to distinguish one kind of contour from another.

c) *Interpreting contours*

Even after contours have been found and labeled, not much is known about the physical structure of the scene. It is clear that contours play an important role in the human perceiver's ability to decide how things are shaped and where they are, apart from the application of specific "higher level" knowledge to objects of known shape.

In this section we study this problem of contour interpretation.

3.2. The Passive Approach (Previous Work)

The recovery of three-dimensional shape and surface orientation from a two-dimensional contour is a fundamental process in any visual system. Recently, a number of methods have been proposed for computing shape from contour. For the most part, previous passive techniques have concentrated on trying to identify a few simple, general constraints and assumptions that are consistent with the nature of all possible objects and imaging geometries in order to recover a single "best" interpretation from among the many that are possible for a given image. For example, Kanade (1981) defines shape constraints in terms of image space regularities such as parallel lines and skew symmetries under orthographic projection. Witkin (1981) looks for the most uniform distribution of tangents to a contour over a set of possible inverse projections in object space under orthography. Similarly, Brady and Yuille (1984) search for the most compact shape (using the measure of area over perimeter squared) in the object space of inverse projected planar contours.

Rather than attempting to maximize some general shape-based evaluation function over the space of possible inverse projective transforms of a given image contour, and adhering to our framework of attempting unique solutions without employing any restrictive assumptions and heuristics, we propose to find a unique solution by using an active approach, since it can be easily proved that one image of a planar contour (under orthography or perspective) admits infinitely many interpretations of the structure of the world plane on which the contour lies, if no other information is known. Finally, the need for a unique solution, which is guaranteed in our approach, arises also from the fact that there exist many real world counterexamples to the evaluation functions that have been developed to date. For example, Kanade's and Witkin's measures incorrectly estimate surface orientation for regular shapes such as ellipses (which are often interpreted as slanted circles). Brady's compactness measure does not correctly interpret non-compact figures such as a rectangle since he will compute it to be a rotated square (e.g. if we view a rectangular table top, we do not see it as a rotated square surface, but as a rotated rectangle). It is worth noting that the equations used in previous work (the passive approach) are highly nonlinear.

Up to now we have only discussed monocular passive shape from contour. It would seem that detecting shape from contour employing a binocular observer might reduce ambiguity and nonlinearity. It is shown in the sequel that this is not the case, i.e., even if we employ a binocular static observer, the problem is still nonlinear, if we want to avoid solving the correspondence problem between the left and right frames. (We have stated that our approach will be correspondenceless.) Indeed, consider a binocular observer imaging a plane contour C with pro-

jections C_L and C_R on the left and right frames respectively. Let a coordinate system $OXYZ$ be fixed with respect to the left camera, with the Z axis pointing along the optical axis. We assume that the image plane I_{m_l} is perpendicular to the Z axis at the point (0,0,1). Let the nodal point of the right camera be the point $(d,0,0)$, and let its image plane I_{m_r} be identical to the previous one. Consider also a plane P in the world with equation $Z = pX + qY + c$, which contains a contour C, and consider the images (perspective) C_l and C_r of the contour on the left and right image planes respectively.

From now on we will denote the coordinates on the left and right image planes by (x_l,y_l) and (x_r,y_r) respectively. We assume perspective projection. We can easily prove that if S_l, S_r are the areas enclosed by contours C_l and C_r and (A_L,B_L), (A_R,B_R) the centers of mass of contours C_l and C_r, then

$$\frac{S_L}{S_R} = \frac{1 - A_L p - B_L q}{1 - A_R p - B_R q}, \tag{3.1}$$

where (p,q) is the gradient of the plane Π with equation $Z = pX + qY + c$ with respect to the left frame, and where we have assumed (for simplicity and without loss of generality) that the focal length $f = 1$. For a proof of (3.1) see Appendix 1. If we want to recover the shape of the contour in view without any assumptions, what we should do is connect properties of the left and right images, if we don't want to resort to correspondence. Such properties can include area, perimeter, any function of these two, or other functions of the positions of the contours. Equation (3.1) is linear and is the only linear constraint we have been able to find.

Another constraint can be extracted from the perimeters of the two contours. If we calculate the perimeter of the world contour from each of two projections, then these two results should be equal. From this we can get an additional constraint on p,q.

To do this, we need to develop the first fundamental form of the world plane as a function of the retinal coordinates, in order to be able to compute the length of the world contour (up to a constant factor, of course). If we fix a coordinate system $OXYZ$ with the Z axis as the optical axis and focal length 1 and we consider a plane Π: $Z = pX + qY + c$ in the world with a contour C on it, and we denote by (x,y) the coordinates on the image plane, then a point (X,Y,Z) in the world planar contour C is projected onto the point

$$x = \frac{X}{Z}; \quad y = \frac{Y}{Z}$$

The inverse imaging function, call it f, is the function that maps the image plane onto the world plane; so, if (x,y) is an image point, the 3-D world point on the plane $Z = pX + qY + c$ that has (x,y) as its image is given by

$$f(x,y) = \left(\frac{cx}{1 - px - qy}, \frac{cy}{1 - px - qy}, \frac{c}{1 - px - qy} \right)$$

The first fundamental form of f [Lipschutz, 1969] is the quadratic form

$$E\,dx^2 + 2F\,dx\,dy + E\,dy^2,$$

with

$E = f_x \cdot f_x$
$F = f_x \cdot f_y$ and
$G = f_y \cdot f_y$.

After simple calculations we get

$$E = \frac{c^2}{(1 - px - qy)^4}\left[(1 = qy)^2 + p^2y^2 + p_2\right]$$

$$F = \frac{c^2}{(1 - px - qy)^4}\left[(1 - qy)qx + (1 - px)py + pq\right]$$

$$G = \frac{c^2}{(1 - ps - qy)^4}\left[q^2x^2 + (1 - px)^2 + q^2\right]$$

So, if we consider two points (x,y) and $(x + dx, y + dy)$ on the image plane, then the three-dimensional distance dC of the corresponding points on the world plane is given by

$$dC = \sqrt{(Edx^2 + 2F\,dx\,dy + G\,dy^2)}$$

Consequently, if we have a contour C on the image plane, then the 3-D planar contour has length

$$\int_C \sqrt{(E\,dx^2 + 2F\,dx\,dy + Gdy^2)}$$

Using the above equation we can compute the length of the world contour (up to a constant factor) from both the left and right frames, and equate the results. This will result in an equation (nonlinear) with unknown p,q. This equation may be solved, together with the linear constraint (Aloimonos, 1986), to give the orientation gradient of the contour. Uniqueness is not guaranteed theoretically, and the nonlinearity could create instabilities.

We now proceed to study the same problem, but using an active observer. We distinguish between a monocular and a binocular observer.

3.3. The Active Approach

3.3.1. Monocular Observer

This problem has already been addressed by Aloimonos (1986) and Kanatani (1986). In Kanatani's scheme, the method developed is an application of the linear feature theory introduced by Amari (1978). The treatment is for differential motion. In Kanatani's scheme, if we have a closed curve C on the image plane, then features are defined as various line integrals along C of the form

$$I = \int_C F(x,y)\,ds,$$

with $ds = \sqrt{dx^2 + dy^2}$, and F any differentiable function. Then, the change $\frac{dI}{dt}$ as the observer moves is connected

through linear equations to the gradient (p,q) of the plane on which the contour lies. In other words, if the observer moves with a known motion, then from the two successive images of the contour, the shape (p,q) of the three-dimensional contour is uniquely computed, if certain conditions are satisfied. The solution is given through linear equations. The sensitivity of the method to noise depends on the error introduced form the numerical differentiation and only on this, and as reported the method seems unstable in the presence of noise.

3.3.2. Binocular Observer

Here we examine the active perception of shape from contour by a binocular observer. Again, let a coordinate system be fixed with respect to the left camera with the Z axis pointing along the optical axis. We consider that the image plane I_{m_l} is perpendicular to the Z axis at the point (0,0,1). Let the nodal point of the right camera be at $(d,0,0)$, and let its image plane I_{m_r} be identical to the previous one. Consider a plane P in the world with equation $Z = pX + qY + c$ that contains a contour C and consider the perspective images C_l and C_r of the contour on the left and right image frames respectively. Let S_L and S_R be the areas of the left and right image contours respectively. Then (see Appendix 1)

$$\frac{S_L}{S_R} = \frac{1 - A_L p - B_L q}{1 - A_R p - B_R q}, \tag{3.2}$$

where (A_L,B_L), (A_R,B_R) are the centers of mass of the left and right contours respectively. We call this constraint the area-ratio constraint. Now, we rotate both eyes by a small angle θ around the X axis (this can also be simulated). The new image coordinates are

$$x' = \frac{r_{11}x + r_{21}y + r_{31}}{r_{13}x + r_{23}y + r_{33}}$$

$$y' = \frac{r_{12}x + r_{22}y + r_{32}}{r_{13}x + r_{23}y + r_{33}}, \quad \text{where}$$

$$R = (r_{ij})_{3\times 3} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}$$

Hence

$$x' = \frac{x}{\cos\theta - y\sin\theta}$$

$$y' = \frac{\sin\theta + y\cos\theta}{\cos\theta - y\sin\theta}$$

Then the new gradient (p',q') of the world contour is

$$p' = \frac{p}{\cos\theta + q\sin\theta} \tag{3.3}$$

$$q' = \frac{q\cos\theta - \sin\theta}{\cos\theta + q\sin\theta} \tag{3.4}$$

But again from the area ratio constraint

$$\frac{S_L^R}{S_R^R} = \frac{1 - A_L^R p' - B_L^R q'}{1 - A_R^R p' - B_R^R q'} \tag{3.5}$$

where (A_L^R,B_L^R), (A_R^R,B_R^R) are the centers of mass of the left and right contours after the rotation and S_L^R, S_R^R are their areas, respectively.

Using (3.3), (3.4) in (3.5) we get

$$\frac{S_L^R}{S_R^R} = \frac{\cos\theta + q\sin\theta - A_L^R p - B_L^R(q\cos\theta - \sin\theta)}{\cos\theta + q\sin\theta - A_R^R p - B_R^R(q\cos\theta - \sin\theta)} \tag{3.6}$$

Equations (3.2) and (3.6) constitute a linear system in the unknowns p and q that has a unique solution in general. These equations, however, become degenerate when $p = 0$; this is because when $p = 0$, both equations reduce to

$$\frac{S_L}{S_R} = \frac{1 - B_L q}{1 - B_R q}.$$

But the y-coordinates are the same in both images, and so $B_L = B_R$ and consequently $S_L = S_R$. So, if $p = 0$ the areas in both images are equal. Appendix 2 develops a condition that proves that this is sufficient too, i.e. $p = 0$ iff the areas in both images are equal. Here, we devise a method for computing q in case $p = 0$.

We can easily prove that if both p and q are zero (world plane parallel to image planes), then the length of the contours in both images (perimeters) are equal. It is not sufficient, though, for the lengths of the contours to be equal. This does not imply that $p = q = 0$; there are some degenerate cases, which are discussed in (Aloimonos and Basu, 1986). For the purposes of this section, assuming that we can check for the degenerate cases, we propose the following algorithm for actively perceiving shape from contour in the case of $p = 0$.

Step 1: Rotate the cameras so that discrepancy between the lengths of the left and right contours is minimized.

Step 2: q corresponds to the rotation that minimizes the discrepancy.

Let $(0,q,-1)$ be the surface normal of the world contour. Rotating the camera by θ around the x-axis, we get

$$\begin{bmatrix} p' \\ q' \\ k' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} 0 \\ q \\ -1 \end{bmatrix} \quad \text{or}$$

$$\begin{bmatrix} p' \\ q' \\ k' \end{bmatrix} = \begin{bmatrix} 0 \\ q\cos\theta + \sin\theta \\ q\sin\theta - \cos\theta \end{bmatrix}.$$

We need $q' = 0$, hence

$$q\cos\theta + \sin\theta = 0 \quad \text{or} \quad q = -\tan\theta.$$

Thus, if θ is the rotation that minimizes the difference between contour lengths, then $q = -\tan\theta$ gives the corresponding q. A similar analysis can be done for a verging stereo system. The mathematics is much more complicated and it can be found in (Aloimonos and Basu, 1986).

3.4. Summary and Discussion

We have presented a theory for the computation of shape from contour by an active observer. The constraints involved demonstrate the superiority of an active observer vs. a passive one, with respect to computation. Uniqueness and linearity vs. ill-posedness and instability make the AV paradigm with respect to the shape from contour problem very appealing and worth studying. The advantage of binocular shape from contour over monocular shape from contour lies in the fact that the monocular case has been demonstrated to be unstable. The problem with the active binocular shape from contour theory is that the contours in the left and right images have to be corresponded. We have not solved this problem but it is certainly easier to correspond macrofeatures (contours) rather than microfeatures. Finally, the problem of shape from nonplanar contour, i.e. understanding the structure of the contour without correspondences in an active way, is treated in (Ito and Aloimonos, 1987a). We chose here the case of a planar contour, because this case has been addressed extensively by past research.

4. SHAPE FROM TEXTURE

The problem of shape from texture has received a lot of attention in the past few years and some excellent research on the topic has been published (Gibson, 1950; Stevens, 1980; Witkin, 1980; Davis et al., 1983; Kanatani, 1984; and Aloimonos, 1986). The problem in the passive case is defined as "finding the orientation of a textured surface from a static monocular view of it." This problem is ill-posed in the sense that there exist infinitely many solutions. To restrict the space of solutions, assumptions have to be made about the texture. Assumptions such as directional isotropy and uniform density have been employed in previous research. Uniform density has been defined as density of texels or density of the sum of the lengths of the contours (zero-crossings) in the image.

It is very clear that even though some of the assumptions used in the literature for the recovery of shape from texture are general enough, they are not powerful enough to capture a very large subset of natural images. As a result, the developed algorithms fail when they are applied to many real surfaces. Furthermore, there is no way to check in advance whether or not a particular assumption is valid for the surface that is imaged. This problem alone is enough to demonstrate the restricted applicability of the existing shape from texture algorithms (or of the ones yet to come). We will show that if the observer is active then the shape from texture problem, or the problem of shape detection from surface intensity and markings, becomes easy, in the sense that no restrictive assumptions are necessary and the solution is obtained from linear equations.

The next section introduces the mathematical prerequisites. For simplicity, and without loss of generality, we will assume that the surface in view is planar (as in previous shape from texture research). If the surface in view is nonplanar, then the problem can be addressed either by applying our theory locally in the image, i.e. assuming that the surface in view is locally planar, or if a parametric model for the surface is assumed, then the same basic principles reported here may be used to recover the parameters of the surface.

4.1. Prerequisites

Suppose that the camera is looking at a planar surface. Assume further that the camera is moving. For our analysis we assume that the surface is moving. This is equivalent to the motion of the camera, and it is done here for simplification of the formulas. Call the planar surface in the world W and the image plane R. Suppose that point $\mathbf{X}=(X,Y,Z)\in W$ is projected onto point $\mathbf{x}=(x,y)\in R$. Let the motion of the surface consist of a translation $\mathbf{T}=(t_1,t_2,t_3)$ and a rotation $\Omega=(\omega_1,\omega_2,\omega_3)$, or $\mathbf{V}(\mathbf{X})=\mathbf{T}+\Omega\times\mathbf{X}$, where $\mathbf{V}(\mathbf{X})$ is the velocity of a point $\mathbf{X}\in W$. Then this velocity can be written as $\mathbf{V}(\mathbf{X})=\sum_{k=1}^{6} r_k\,\mathbf{V}_k(\mathbf{X})$, where

$$r_1=t_1,\quad V_1(\mathbf{x})=(1\ 0\ 0)^T$$

$$r_2=t_2,\quad V_2(\mathbf{x})=(0\ 1\ 0)^T$$

$$r_3=t_3,\quad V_3(\mathbf{x})=(0\ 0\ 1)^T$$

$$r_4=\omega_1,\quad V_4(\mathbf{x})=(0\ -Z\ \ Y)^T$$

$$r_5=\omega_2,\quad V_5(\mathbf{x})=(Z\ 0\ -X)^T$$

$$r_6=\omega_3,\quad V_6(\mathbf{x})=(-Y\ \ X\ 0)^T$$

Then, it can be easily proved that the optic flow (image velocity) at a point $\mathbf{x}=(x,y)$ is $\dot{\mathbf{x}}=\sum_{k=1}^{6} r_k\,\mathbf{u}_k(\mathbf{x})$.

We prove the above equation avoiding the details of the perspective projection. Let the projection from the world to the image plane be $\mathbf{P}$, with $\mathbf{P}(\mathbf{X})=\mathbf{x}=(x,y)=\left(\mathbf{P}_1(\mathbf{X}),\mathbf{P}_2(\mathbf{X})\right)$. If the shape of the surface W is given (a function h), a mapping: $P_h^{-1}: R\rightarrow$ object surface is defined such that $\mathbf{P}\left(P_h^{-1}(\mathbf{x})\right)=\mathbf{X}$.

The optic flow $\dot{\mathbf{x}}$ at a point $\mathbf{x}=(x,y)$ is then given by

$$\dot{\mathbf{x}}=\frac{d}{dt}\mathbf{P}(\mathbf{X})=\frac{\partial\mathbf{P}}{\partial\mathbf{X}}\mathbf{V},$$

where both $\frac{\partial\mathbf{P}}{\partial\mathbf{X}}$ and $\mathbf{V}$ are functions of retinal coordinates.

So, since $\mathbf{V}=\sum_{k=1}^{6} r_k\mathbf{V}_k$, we have

$$\dot{\mathbf{x}}=\sum_{k=1}^{6} r_k\,\mathbf{u}_k\,(\mathbf{x}) \text{ with} \tag{4.1}$$

$$\mathbf{u}_k\,(\mathbf{x})=\frac{\partial\mathbf{P}}{\partial\mathbf{x}}\left(P_h^{-1}(\mathbf{x})\right)\,\mathbf{V}_k\,\left(P_h^{-1}(\mathbf{x})\right).$$

Equation (4.1) will be used very frequently in the sequel.

4.2. Linear Features

Here we introduce the concept of a *linear feature vector* that has proved to be a strong device for several problems in cybernetics (Amari, 1978). Let the image intensity function be denoted by $s(x,y)$. A *linear feature* (LF) is a linear function f over the image, i.e.

$$f = \iint s(x,y)\, m(x,y)\, dx\, dy\,,$$

where m is called a measuring function, s is the image brightness and the integration is taken over the area of interest. A *linear feature vector* $\mathbf{f}$ (LFV) is a vector of linear features, i.e.

$$\mathbf{f} = [f_1 f_2 \cdots f_n]^T\,, \text{ with}$$

$$f_i = \int\int s\, m_i\, dx\, dy$$

where m_i is a measuring function, for $i = 1, \ldots, n$. $\{m_i\}$ could be any set; one good example is $\{m_i\} = \{mp(\} = \{e^{i(x+qy)}\}$, in which case a linear feature corresponds to a Fourier component of the image.

4.3. The Constraint

Since there is motion, the induced optical flow satisfies the following equation (approximately):

$$s_x\, u + s_y\, v + s_t = 0,$$

where (u,v) is the optic flow at a point (x,y) and s_x, s_y, s_t are the spatiotemporal derivatives of the image intensity function at the point (x,y). This equation can be written as

$$\frac{\partial s}{\partial t} = -\dot{\mathbf{x}} \cdot \nabla s.$$

The time derivative of an LFV will be

$$\dot{\mathbf{f}} = [\dot{f}_1 \dot{f}_2 \cdots \dot{f}_n], \text{ where}$$

$$\dot{f}_i = \int\int \frac{\partial s}{\partial t}\, m_i\, dx\, dy = -\iint m_i\, (\dot{\mathbf{x}} \cdot \nabla s)\, dx\, dy$$

The optic flow field (from equation 4.1) can be written in the form

$$\dot{\mathbf{x}} = \sum_{k=1}^{6} r_k\, \mathbf{u}_k = \sum_{k=1}^{6} r_k \begin{bmatrix} u_k(\mathbf{x}) \\ v_k(\mathbf{x}) \end{bmatrix}$$

From this,

$$\dot{f}_i = -\int\int m_i\, (\dot{\mathbf{x}} \cdot \nabla s)\, dx\, dy \text{ or}$$

$$\dot{f}_i = -\sum_{k=1}^{6} r_k \iint m_i\, (u_k\, s_x + v_k\, s_y)\, dx\, dy \text{ or}$$

$$\dot{f}_i = \sum_{k=1}^{6} r_k\, h_{ik}\,, \text{ with}$$

$$h_{ik} = -\iint m_i\, (u_k\, s_x + v_k\, s_y)\, dx\, dy$$

So, we have found that

$$\dot{\mathbf{f}} = H \cdot r, \qquad (4.2)$$

where $H = (h_{ik})$ and $r = (r_1 r_2 \cdots r_6)^T$, the motion parameters.

Matrix H contains the parameters of the plane in a linear form. So, equation (4.2) relates linear features with shape and motion parameters. Furthermore, it is linear in the shape of the planar surface in view. So, a simple linear least-squares method or a Hough transform technique is sufficient for the recovery of the gradient of the plane in view. Depth can be computed too, if desired.

Finally, we want to stress here the fact that in this algorithm, the spatial derivatives of the intensity function don't need to be computed. This is due to the linear feature vector approach. (Integration by parts avoids differentiation of the intensity function. Instead, the derivative of the measuring function has to be computed. So, we avoid differentiating the image intensity, which is discrete, because numerical differentiation is an ill-posed problem.) More importantly, the same approach can be followed if the image is a dot pattern (or a line pattern—zero crossings), i.e. it is discontinuous. The reason for this is again the fact that the spatial derivatives of the intensity function don't have to be estimated. Only the temporal derivative of the image needs be estimated. This approach has been initiated in (Ito and Aloimonos, 1987).

4.4. Summary and Discussion

We have presented a method for the recovery of the shape of a planar surface by an active observer. Our method does not rely on any assumptions about the texture and it does not require the image to be spatially differentiable. The approach is based on the fact that the observer is moving with a known motion. If the observer is moving with an unknown motion, then again the problem is solvable, and it has been addressed by Aloimonos and Bandyopadhyay (1986), Negahdaripour (1986), and Amari et al. (1985). We will report elsewhere our research on this case.

5. STRUCTURE FROM MOTION

5.1. Introduction

The problem of structure from motion has received considerable attention lately (Ullman, 1979; Longuet-Higgins and Prazdny, 1980; Tsai and Huang, 1984). The problem is to recover the three-dimensional motion and structure of a moving object from a sequence of its images. Even though computation of structure and 3-D motion are equivalent when the retinal motion is given, the two problems have received different names, the former "Structure from Motion" and the latter "Passive Navigation." We will refer to them interchangeably.

Basically there have been two approaches toward solving this problem. The first assumes "small" motion. In this case, if the three-dimensional intensity function (two spatial and one temporal argument) is locally well-behaved and its spatiotemporal gradients are defined, then the image velocity field (or optic flow) may be computed (Horn and Schunck, 1981; Hildreth, 1984). Algo-

rithms developed using this approach use the velocity field to compute 3-D motion and structure. The second approach assumes that the motion is large and measurement of image motion entails solving the correspondence problem. Imaged feature points due to the same three-dimensional artifact (e.g. texture element or edge junction) in two successive dynamic frames are assumed to be identified correctly. Algorithms using this approach compute 3-D transformation parameters from the above mentioned displacements field (Tsai and Huang, 1984; Roach and Aggarwal, 1980; Nagel and Neumann, 1981). There is a third approach, that computes 3-D motion directly from brightness patterns, but the general case (unrestricted motion) has not been solved yet (Aloimonos and Brown, 1984; Negahdaripour and Horn, 1985).

The "small" (continuous) and "large" (discrete) motion cases are slightly different in terms of the constraints that relate the 3-D to the 2-D motion, although the results are essentially the same (Fang and Huang, 1984). So, we concentrate here only on the continuous case, where the input to the "structure from motion" perceptual process is the optic flow field. What will be developed in the sequel has meaning if and only if the optic flow (image velocity) is the projection of the three-dimensional motion. We state this explicitly, because the velocity of the brightness patterns in the image is computed (according to the existing literature) using some assumptions, which might violate the fact that image velocity is the projection of 3-D motion. In the case where optic flow is used as input, there is some excellent research (Bruss and Horn, 1983; Prazdny, 1981). The basic problems of this passive approach are:

1) The constraint that relates 3-D to 2-D motion is nonlinear.

2) The dimensionality of the space of unknows is high (five, if one camera is used).

Sometimes, closed form solutions may be found (Longuet-Higgins and Prazdny, 1980), but higher order derivatives of the optic flow are involved. Thus, given that optic flow will be noisy (there is no algorithm to date that can compute optic flow in natural scenes with high accuracy) and also given that numerical differentiation is an ill-posed problem (Poggio et al., 1986), the efficacy of these approaches is questionable. In this section we prove that an active observer solves the structure from motion problem more efficiently. In particular, an active monocular observer brings down the dimension of the space of the unknowns from five to four, but he does not get rid of the nonlinearity. An active binocular observer greatly simplifies the constraints used in the analysis of motion and permits simple closed form solutions of the resulting parameter equations.

The strategy advocated in this section calls for visual tracking and here the formulation is the one in (Bandyopadhyay, 1986). This approach is suggested as a possible way to overcome the difficulties of the passive monocular approach. The possible employment of active tracking to facilitate navigation has been suggested by visual psychologists (Cutting, 1982).

It will be shown (Bandyopadhyay, 1986) that when the observer is able to track a prominent feature point in the imaged scene, the task of navigation is facilitated since it is easier to compute egomotion parameters, compared to the non-tracking case. The emphasis in this section is on the mathematics governing the imaging equations that are obtained while the system is tracking. To track, the system must have some way of measuring the error in the retinal signal.
The outline of this section is as follows:

1. Error velocity measurement to correct tracking drift is discussed in light of the primate pursuit system.
2. A general form of the relation between 3D velocity parameters and retinal optical flow is derived. In previous derivations of this relation, the origins of the body centered coordinate frame and viewer centered coordinate frame are taken to coincide at the instant of measurement. Using the general representation it is shown why a monocular observer, who is able to track an environmental feature point, has to contend with a smaller number of velocity parameters.
3. A new set of constraint equations are derived for the tracking observer, which allow closed form solution of the egomotion parameters. Simulation results are described and implementational issues for integrating this module into the overall motion interpretation scheme are discussed.

5.2. Target Selection Via Velocity Channels

The key assumption is that the alignment of the camera axes is controllable by the system itself. In this case, as the system moves in the world, the orientation of the camera is continually adjusted. This adjustment is dependent upon the two dimensional motion perceived on the retina.

In the tracking system the problem can be seen as: given the image of a target environmental point, to generate control signals that will *foveate* the target. The block diagram of a system for accomplishing this can be schematized as shown in Figure 5.1. The first and most important point to make is that the system can be adequately modeled by servomechanism concepts. It is relatively easy to see how to generate the kinds of motor commands for the two movement system to produce the observed behavior. This of course assumes that the target point is identified.

Target identification is a central issue: in a complicated motion field, how can the target velocities be easily identified? This is a basic subproblem in tracking using velocity sensing and is captured by Figure 5.2.

Our answer to this question uses the notion of global flow field vectors. Such vectors respond to velocities in every part of the optical flow field. In other words, if we visualize the optic flow field as a four dimensional parameter space $(x, y, u(x, y), v(x, y))$, the global flow field

sums all the different flow vectors in a two dimensional (u, v) parameter space. The detectors' sensitivities are organized into channels. In the case of a particular flow field, some channels will typically respond to it and others will not. Figure 5.3 shows how the channel concept can be utilized.

We claim that with this abstract flow channel model the problem becomes one of determining which of the channels should be used for the eye movement control system. This means that a mechanism is needed to switch the appropriate channels into the servo system. Note that this technique uses a spatially distributed detector array. Our contention is that it is appropriate to average the flow field over this subset.

A mechanism to switch the detectors on once the appropriate ones have been identified is simple to understand, so we will concentrate on identifying the ideas behind selecting the right detectors. The general way that this is done is by a feed-forward mechanism that determines some selection criterion. The different kinds of criteria are important, so it is useful to categorize them.

1. *Extrinsic features.* This method uses some other feature, say color, that also has spatially organized detectors. To track a red object, the detectors that register red are used to select the spatial component of the velocity detectors. All such detectors with the appropriate correspondence are used.
2. *Intrinsic features.* This method uses some particular range of values for the flow field itself, say all values over a certain velocity magnitude. To track an object, all the detectors that satisfy the intrinsic criterion are switched into the movement control system.

These distinctions are important as they correspond to two different types of tracking situations. In navigation, where the entire spatial field is moving, an extrinsic feature is appropriate. In pursuing a small target, that target is usually moving differently with respect to the background, so an intrinsic feature may be appropriate.

5.3. Measuring Egomotion

5.3.1. Background

Consider first the monocular imaging situation where a sensor is moving relative to a static scene. The coordinate frame (X,Y,Z) is fixed to the sensor (see Figure 5.4). The viewing direction is along the positive Z-axis.

The analysis presented here assumes a rotating and translating observer moving in a static environment. However, since the velocity parameters characterize motion relative to the observer's frame of reference, the analysis per se is not affected by multiple moving objects. The analysis assumes the velocity representation for the motion parameters.

The reference coordinate frame is fixed to the observer. There is another coordinate frame fixed at the point S on the body (see Figure 5.4). The point S has the velocity $T_S = (U_S, V_S, W_s)$. At the time of observation the reference and the body frame axes are parallel to each other. The rotational velocity of the body is given by the vector $\Omega = (\alpha,\beta,\gamma)$. The 3D velocity of a point $P = (X,Y,Z)$ on the body is given by the equation

$$\dot{X} = T + [R](X - X_S) \tag{5.1}$$

where $X_S = (X_S, Y_S, Z_S)$ denote the position of the body origin S, and $\dot{X}$ denotes the 3D velocity of P (the 'dot' operator is used throughout to signify differentiation with respect to time); also

$$[R] = \begin{bmatrix} 0 & -\gamma & \beta \\ \gamma & 0 & -\alpha \\ -\beta & \alpha & 0 \end{bmatrix}$$

Image formation is modeled by perspective projection. The projection of a point $P = (X,Y,Z)$ is denoted by $p = (x,y)$. The projective relation is

$$(x,y) = \left(\frac{fX}{Z}, \frac{fY}{Z}\right) \tag{5.2}$$

The constant f is the focal length of the imaging system. It is the distance separating the nodal point of the camera (or eye) and the image plane, moving along the optical axis (i.e. X axis). In subsequent steps the constant f is assumed to be unity. The velocity of image points in the 2d image space is called optical flow. The relations between the 2D and 3D velocities are obtained by differentiating the equation (5.2) and substituting from equation (5.1):

$$\begin{aligned} u = \dot{x} &= \frac{U_S - xW_S}{Z} - \alpha\left[xy - x\frac{Y_S}{Z}\right] \\ &\quad + \beta\left[1 - \frac{Z_S}{Z} + x^2 - x\frac{X_S}{Z}\right] - \gamma\left[y - \frac{Y_S}{Z}\right] \\ v = \dot{y} &= \frac{V_S - yW_S}{Z} - \alpha\left[1 - \frac{Z_S}{Z} + y^2 - y\frac{Y_S}{Z}\right] \\ &\quad + \beta\left[xy - y\frac{X_S}{Z}\right] + \gamma\left[x - \frac{X_S}{z}\right] \end{aligned} \tag{5.3}$$

When the origin of the body coordinate frame coincides with the reference or observer coordinate frame then $X_S = Y_S = Z_S = 0$, and $T = T_0 = (U,V,W)$, which simplifies the equation for optical flow to give

$$u = \frac{U - xW}{Z} - \alpha xy + \beta(x^2 + 1) - \gamma y \tag{5.4.1}$$

$$v = \frac{V - yW}{Z} - \alpha(1 + y^2) + \beta xy + \gamma \tag{5.4.2}$$

The above pair of equations embodies the constraint that the optical flow (u,v) imposes upon the parameters of rigid motion. Thus all an observer has to do to determine where he is going is to measure the retinal velocity pattern and then use the above pair of equations applied at least five points to determine the 3D velocity of egomotion. Note that there are six velocity components (i.e. three for translation and three for rotation). However, all six parameters cannot be computed by monocu-

lar visual data. This is because of the depth term Z that occurs in the above pair of equations. The depth introduces a scaling effect, whereby other things being equal, multiplying the translational components and the depth by the same constant factor leaves the perceived retinal motion unchanged. Thus for example an object at a certain distance translating with a certain speed generates the same optical flow field when it is twice as far away and travelling in the same direction with twice the speed.

The monocular observer, lacking depth information, must eliminate the depth factor from the optical flow constraints. This will then imply that the observer's translation can only be determined up to a scale factor. Thus the number of egomotion parameters of interest are five—pertaining to the direction of translation and the rotation.

When the depth variable is eliminated from the above equations we have

$$\frac{x_0 - x}{y_0 - y} = \frac{u + \alpha xy - \beta(x^2+1) + \gamma y}{v + \alpha(y^2+1) - \beta xy - \gamma x} \tag{5.5}$$

where $(x_0,y_0) = \left(\frac{U}{W},\frac{V}{W}\right)$ represents the direction of translation of the observer's coordinate frame.

The above constraint equation demonstrates the difficulty of motion computation for a monocular observer. It is nonlinear as well of high dimensionality; these two properties in conjunction make the problem difficult (Tsai and Huang, 1984).

5.3.2. The Tracking Advantage

It will now be shown that in case the monocular observer can discern a distinguishing feature or mark on the observed surface then the perception problem becomes simpler. Suppose that the surface in view has an easily distinguishable and localized feature at point S whose corresponding image location is (x_S,y_S). In this case we can shift the body origin to the point S and rewrite the optical flow equations as in (5.3). In addition

$$\begin{aligned} u(x_S,y_S) = u_S &= \frac{U_S - x_S W_S}{Z_S} \\ v(x_X,y_S) = v_X &= \frac{V_S - y_S W_S}{Z_S} \end{aligned} \tag{5.6}$$

Combining equations (5.6) and (5.3) one obtains

$$u = \frac{u_S + (x_S - x)W_S'}{Z'} + \frac{\alpha x y_S - \beta(1 + x x_S) + \gamma y_S}{Z'} - \alpha xy + \beta(1+x^2) - \gamma y \tag{5.7.1}$$

$$v = \frac{v_S + (y_S - y)W_S'}{Z'} + \frac{\alpha(y y_S + 1) - \beta x_S y - \gamma x_S}{Z'} - \alpha(1+y^2) + \beta xy + \gamma x \tag{5.7.2}$$

where the prime signifies scaling by Z_S, i.e. $W_S' = \frac{W_s'}{Z_s}$.

Note that the translational parameters with respect to the observer's frame (i.e. the observer's actual translation) are related to the body centered translational parameters by

$$\begin{aligned} U' &= U_S' - \beta + \gamma y_S \\ V' &= V_S' + \alpha - \gamma x_S \\ W' &= W_S' - \alpha y_S + \beta x_S \end{aligned} \tag{5.8}$$

The above analysis illustrates the fact that given the ability to estimate the projected velocity of a localized feature accurately, the constraint equations reduce in dimensionality by one.

A similar result may be obtained, as can be expected, when the moving observer is able to track a single feature point so that it appears stationary on the retina at position (0,0). In this case we assume that the tracking motion consists of rotations about the axes that are orthogonal to the line of sight or the optical axis of the lens. The tracking motion is a rotation $(\omega_x,\omega_y,0)$, which is superimposed upon the actual parameters of motion.

Let $S = (0,0,Z_0)$ be the spatial coordinates of the point being tracked. Assume that the observer can track an environmental point and hold it steady on the optical axis (Z axis). Therefore the optical flow field will have a singularity at the origin of the retinal frame, where the flow value is zero. At the time of observation, the tracked point tends to move along the observer's optical axis (Figure 5.5).

Consider an observer moving with translation (U,V,W) and rotation (α,β,γ). Then, if the body frame origin is taken to be at S, from equation (5.8), remembering that $U_S = V_S = 0$:

$$\begin{aligned} U' &= \frac{U}{Z_0} = -B \\ V' &= \frac{V}{Z_0} = A \\ W_S &= W \end{aligned} \tag{5.9}$$

Furthermore, the optical flow equation (5.3) becomes

$$\begin{aligned} u &= \frac{-xW}{Z} - Axy + B\left[1 - \frac{Z_0}{Z} + x^2\right] - \gamma y \\ v &= \frac{-yW}{Z} - A\left[1 - \frac{Z_0}{Z} + y^2\right] + Bxy + \gamma x \end{aligned} \tag{5.10}$$

where $A = \alpha + \omega_x$ and $B = \beta + \omega_y$.

Eliminating Z from the above we have

$$\frac{u + Axy - B(x^2+1) + \gamma y}{v + A(1+y^2) - Bxy - \gamma x} = -\frac{B + xW'}{A - yW'} \tag{5.11}$$

where $W' = \frac{W}{Z_0}$.

The constraint equations derived above are similar in form to equation (5.5). However, in this case the dimensionality of the parameter space has been reduced from

five to four *without increase in the degree of nonlinearity of the constraint.* It is important to note that the observer can determine his direction of translation since from equation (5.9) we have

$$U' = \frac{U}{Z_0} = -B$$
$$V' = \frac{V}{Z_0} = A$$

Thus even without explicitly measuring his tracking motion, the observer can determine the scaled translation (U', V', W').

5.4. Binocular Tracking

The optical axes of the two cameras converge onto a point in the environment that is being tracked. The geometry is illustrated in Figure 5.7. It will be shown that the tracking velocities, which are assumed to be observable, can be used to compute egomotion parameters. We will also assume a rectilinearly moving observer, with negligible instantaneous acceleration rates. The analysis generally deals with the left coordinate frame, with respect to which the various quantities will be written as in the monocular case. When it is needed to reference the quantities with respect to the right frame these will be written primed (e.g. x'). The tracking motion involves three independent rotational velocities $(\omega, \omega_y, \omega_y')$. The rotation ω is about the baseline RL of the imaging system (Figure 5.7). Hence the tracking motion of the left frame is given by $(\omega_x = -\omega\sin\theta, \omega_y, \omega_z = \omega\cos\theta)$. If Z_0 is the depth of the tracked point in the left frame then

$$\frac{2d}{\sin(\theta+\theta')} = \frac{Z_0}{\sin(\theta')} = \frac{Z_0'}{\sin(\theta)}$$

Thus we can write

$$Z_0(t) = 2d\frac{\sin(\theta')}{\sin(\theta+\theta')} = F[\theta(t), \theta'(t)]$$

Also

$$\dot{F} = 2d\left[\frac{\dot{\theta}'\cos\theta'\sin(\theta+\theta') - (\dot{\theta}+\dot{\theta}')\sin\theta'\cos(\theta+\theta')}{\sin^2(\theta+\theta')}\right]$$

which simplifies to

$$\dot{F} = 2d\left[\frac{\dot{\theta}'\sin\theta - \dot{\theta}\sin\theta'\cos(\theta+\theta')}{\sin^2(\theta+\theta')}\right] \qquad (5.15)$$

Differentiating the above relation with respect to time once more, we have

$$\ddot{F} = 2d\left[\frac{\ddot{\theta}'\cos\theta'}{\sin(\theta+\theta')} - \frac{(\ddot{\theta}+\ddot{\theta}')\sin\theta'\cos(\theta+\theta')}{\sin^2(\theta+\theta')}\right]$$
$$+ 2d\left[\frac{(\dot{\theta}+\dot{\theta}')^2\sin\theta' - (\dot{\theta}')^2\sin\theta'}{\sin(\theta+\theta')}\right]$$
$$- 4d\left[\frac{(\dot{\theta}+\dot{\theta}')\dot{\theta}'\cos\theta'\cos(\theta+\theta')}{\sin^2(\theta+\theta')} - \frac{(\dot{\theta}+\dot{\theta}')^2\sin\theta'\cos(\theta+\theta')}{\sin^3(\theta+\theta')}\right]$$

Simplifying the above leads to

$$\ddot{F} = 2d\left[\frac{\ddot{\theta}'\sin\theta - \ddot{\theta}\sin\theta'\cos(\theta+\theta')}{\sin^2(\theta+\theta')}\right]$$
$$+ 2d\left[\frac{\dot{\theta}(\dot{\theta}+2\dot{\theta}')\sin\theta'}{\sin(\theta+\theta')}\right]$$
$$- 2(\dot{\theta}+\dot{\theta}')\cos(\theta+\theta')\dot{F} \qquad (5.16)$$

Let the motion of the observer be described by the translational velocity $T = (U, V, W)$ and rotational velocity $\Omega = (\alpha, \beta, \gamma)$. These parameters are defined with respect to the point L in the body, which also happens to be the origin of the left coordinate frame. The tracking motion of the system consists of three independent rotations with respect to the observer. These three rotations correspond to the three motors in Figure 5.1. The angular velocity ω corresponds to the rotation of the plane PRL about the axis LR. The other two angular velocities are $\dot{\theta}$ and $\dot{\theta}'$, which affect the left and right coordinate frames respectively. Let the sense of ω be positive in the direction from L to R. Then the tracking angular motion of the left frame with respect to the observer is given by ω_t and that of the right frame is ω_t'. Note that we will the express all the motion parameters measured in a frame with respect to basis vectors defined in that frame. Therefore

$$\omega_t = (-\omega\sin\theta, \dot{\theta}, \omega\cos\theta)$$
$$\omega_t' = (-\omega\sin\theta', \dot{\theta}', -\omega\cos\theta')$$

If the rigid motion parameters with respect to the right coordinate frame are given by the translational velocity T' and the rotational velocity Ω' then we have

$$T' = R_\lambda \cdot (T + \Omega \times \rho)$$
$$\Omega' = R_\lambda \cdot \Omega$$

where $\times$ denotes the vector product and $\cdot$ denotes matrix multiplication. In addition, the rotation matrix R_λ expresses the transformation due to the rotation by $\lambda = \pi - (\theta + \theta')$ between the left and right frames and is

$$R_\lambda = \begin{bmatrix} \cos\lambda & 0 & -\sin\lambda \\ 0 & 1 & 0 \\ \sin\lambda & 0 & \cos\lambda \end{bmatrix}$$

Now from equation (5.9) we have

$$\dot{F}(t) = W$$
$$U + (\beta + \omega_y)F(t) = 0 \qquad (5.17)$$
$$V - (\alpha + \omega_x)F(t) = 0$$

Observe that the above equations involve five unknown motion parameters. If we now differentiate these equations we have

$$\ddot{F}(t) = \dot{W}$$
$$\dot{U} + \beta\dot{F}(t) + \dot{\beta}F(t) + \omega_y F(t) + \omega_y \dot{F}(t) = 0 \qquad (5.18)$$
$$\dot{V} - \dot{\alpha}F(t) - \alpha\dot{F}(t) - \dot{\omega}_x F(t) - \omega_x \dot{F}(t) = 0$$

Although here we consider a rectilinearly moving observer, the translational velocity (U,V,W) undergoes change due to the rotation of the frame in which the observations are made. Thus we obtain

$$\dot{U} = (\beta + \omega_y)W - (\gamma + \omega_z)V$$
$$\dot{V} = (\alpha + \omega_z)W + (\gamma + \omega_z)U$$
$$\dot{W} = (\alpha + \omega_z)V - (\beta + \omega_y)U$$

Similarly the rotational velocity (α,β,γ), undergoes change due to the tracking motion, as follows:

$$\dot{\alpha} = \omega_y\gamma - \omega_z\beta$$
$$\dot{\beta} = -\omega_x\gamma + \omega_z\alpha$$
$$\dot{\gamma} = 0$$

Introducing the parameters $A = \alpha + \omega_x$, $B = \beta + \omega_y$ and $C = \gamma + \omega_z$, substituting for $\dot{U}$, $\dot{V}$, $\dot{W}$, $\dot{\alpha}$ and $\dot{\beta}$ from the above relations, and replacing U and V from (5.17), we have, from the last two equations in (5.18)

$$C = \frac{2B\dot{F}(t) + A\omega_z F(t) + \dot{\omega}_y F(t)}{(A + \omega_z)F(t)}$$

and

$$C = \frac{2A\dot{F}(t) - B\omega_z F(t) + \dot{\omega}_x F(t)}{-(B + \omega_y)F(t)}$$

Finally eliminating C from the above pair of equations and using the remaining equation of (5.18) we obtain the pair of independent equations

$$A^2 + B^2 = \frac{\ddot{F}(t)}{F(t)} = \phi_1 \qquad (5.19.1)$$

$$\phi_2 A + \phi_3 B + \phi_4 = 0 \qquad (5.19.2)$$

where

$$\phi_2 = 2\omega_x\dot{F}(t)F(t) + \dot{\omega}_x F^2(t) + \omega_y\omega_z F^2(t)$$
$$\phi_3 = 2\omega_y\dot{F}(t)F(t) + \dot{\omega}_y F^2(t) - \omega_x\omega_z F^2(t)$$
$$\phi_4 = (\omega_x\dot{\omega}_x + \omega_y\dot{\omega}_y)F^2(t) + \omega_z F(t)\ddot{F}(t)$$

From equation (5.19) we obtain two sets of solutions for the motion parameters. Eliminating the parameter B we have

$$aA^2 + bA + c = 0 \qquad (5.20)$$

where $a = \phi_2^2 + \phi_3^2$, $b = 2\phi_2\phi_4$ and $c = \phi_4^2 - \phi_1\phi_3^2$.

To summarize, the solution method consists of obtaining the solutions to the pair of equations (5.19.1) and (5.19.2). Since closed form solutions are obtained at every time instant and assuming the computation errors to be uniformly random, we perform smoothing on the time series of the computed parameters, to eliminate a large portion of the error.

The important aspects of this method of computation of the motion parameters are as follows:

(a) The solution is in closed form, requiring no iteration or search.

(b) The constraints are derived from the observed tracking velocities and rotations. We do not need the optical flow measurements.

(c) Here the observables are, $(\theta, \theta', \theta, \theta', \theta, \theta')$. These can be measured quite accurately by analog measurement apparatus. This possibility forms a strong motivation for the tracking approach.

(d) The optical flow field in our motion perception scheme is only used to disambiguate between the possible interpretations computed by the tracking module. This is always possible since under extended periods of observation the optical flow field generated is compatible with one and only one interpretation (Bandyopadhyay 1986).

Simulation experiments based on the above analysis show the approach to be robust against noise. The results are illustrated in Figure 5.9, where one of the rotational parameters is plotted against time. The values plotted are those obtained after applying a temporal smoothing filter to the solution of equation (5.20). These parameter values are suitable for use as an initial estimator in a cooperative optical flow and structure computation algorithm. Basically, the optic flow and structure computation are strongly interdependent. An independent computational mechanism for motion parameter estimation can guide the motion correspondence process and enable the computation of structure. Namely, in our method, motion parameters are computed without using optical flow, and then these parameters along with the intensity profile enable us to uniquely compute optic flow which in an unrestricted case is essential for the computation of structure.

6. CONCLUSIONS AND FUTURE RESEARCH

We have proposed a new paradigm for visual perception called active vision. The idea of using active vision to address the structure from motion problem was introduced by Bandyopadhyay (1986). Here we have addressed several other problems such as shape from shading, texture, contour and depth computation. Our methodology was demonstrated by showing its applicability in these important areas of low and intermediate level vision. It was shown that the controlled alteration of viewing parameters yields stable and robust algorithms for shape and motion perception.

The basis for the approach lies in being able to work in a rich stimulus domain with a partially known parametrization. This knowledge is due to the fact that the viewing transformation is known. As the viewing parameters are continuously varied, the observed visual stimuli undergo local transformations that are measurable and provide powerful constraints for the computation of the unknown scene parameters. We are, of course, interested in the rates of these stimulus changes, which have traditionally been thought to be difficult to measure. However, it should be pointed out that in the present scheme we do not work with a small set of discrete obser-

vations, but with trajectories in the stimulus space, termed flow lines. These trajectories are smooth, since the viewing transformations we use are themselves smooth, and therefore can be computed accurately enough for our purposes. Thus we do not need to rely on the smoothness of properties of the observed scene, such as illumination and depth. The real power of the method is due to the avoidance of complications usually associated with multiview approaches to visual perception. For instance, the problem of correspondence of microfeatures is not involved in the current approach.

The treatment in this paper has been largely theoretical, because we want to create a sound framework for what we believe is a promising methodology for computer vision. We plan to verify the analysis with experiments on synthetic and natural images. Research going on currently at the University of Rochester (Brown, 1986) and at SUNY Stony Brook (Bandyopadhyay, 1987) aims toward developing an active vision system that will be able to track objects in real time. On the other hand, we plan to continue our theoretical analysis of active visual computations, before we build a system with possibly special hardware that will carry out active visual computations. The initial developments of the theory of active vision have been outlined. There are many important issues that need to be explored in this context. For instance, we have not looked at the question of whether there is any viewing parameter trajectory that is preferable to other trajectories in tackling the computational task. This could be termed exploratory visual computation, where the knowledge about scene constraints available at any stage of the visual process determines the way in which the control parameters will be altered. Also, a theoretical error analysis of the proposed algorithms has to be done, taking into account discretization effects and noise in images. Finally, the problem of learning the viewing parameter trajectory that is the best (in computational terms) with respect to a particular problem, using a neural (connectionist) network, has to be addressed. Examination of such issues forms an important future research goal.

REFERENCES

Aloimonos, J., "Computing intrinsic images", Ph.D. thesis, University of Rochester, 1986.

Aloimonos, J. and Bandyopadhyay, A., "Correspondence is not necessary for motion perception", submitted.

Aloimonos, J. and Basu, A., "Shape from contour", submitted, 1986.

Aloimonos, J. and Brown, C.M., "Direct processing of curvilinear sensor motion from a sequence of perspective images", *IEEE Workshop on Computer Vision*, Annapolis, MD, 72–77, 1984.

Amari, S., "Feature spaces which admit and detect invariant signal transformations", *Proc. ICPR*, Kyoto, Japan, 452–456, 1978.

Amari, S. and Maruyama, S., "Computation of structure from motion", personal communication, 1986.

Bandyopadhyay, A., personal communication, 1987.

Bandyopadhyay, A., "A computational study of rigid motion perception", Ph.D. thesis, Department of Computer Science, University of Rochester, 1986.

Brady, J. and Yuille, A., "An extremum principle for shape from contour", *IEEE Trans. on PAMI*, **6**, 288–301, 1984.

Brown, C.M., personal communication, 1986.

Bruss, A. and Horn, B.K.P., "Passive navigation", *Computer Vision, Graphics and Image Processing*, **21**, 3–20, 1983.

Cutting, J. E., "Motion parallax and visual flow: how to determine direction of locomotion", Dept. of Psychology, Cornell University, 1982.

Davis, L., Janos, L. and Dunn, S., "Efficient recovery of shape from texture", TR-1133, Computer Vision Laboratory, University of Maryland, 1982.

Fang, J.Q. and Huang, T.S., "Solving three dimensional small rotation motion equations: uniqueness, algorithms, and numerical results", *Computer Vision, Graphics and Image Processing*, **26**, 183–206, 1984.

Gibson, J.J., *The Perception of the Visual World*, Houghton Mifflin, Boston, 1950.

Hadamard, J., *Lectures on the Cauchy Problem in Linear Partial Differential Equations*, New Haven: Yale University Press.

Hildreth, E.C., "Computations underlying the measurement of visual motion", *Artificial Intelligence*, **23**, 309–354, 1984.

Horn, B.K.P., *Robot Vision*, McGraw-Hill, 1986.

Horn, B.K.P., "Understanding image intensities", *Artificial Intelligence*, **8**, 201–231, 1977.

Horn, B.K.P and Schunck, B., "Determining optical flow", *Artificial Intelligence*, **17**, 185–204, 1981.

Ikeuchi, K. and Horn, B.K.P., "Numerical shape from shading and occluding boundaries", *Artificial Intelligence*, **17**, 141–184, 1981.

Ito, E. and Aloimonos, J., "Determining transformation parameters from images: theory", to appear in *Proc. IEEE Conference on Robotics and Automation*, 1987.

Ito, E. and Aloimonos, J., "Shape from nonplanar contour", to appear, 1987a.

Kanade, T., "Recovery of the shape of an object from a single view", *Artificial Intelligence*, **17**, 409–460, 1981.

Kanatani, K., "Group theoretical methods in image understanding", TR-1692, Computer Vision Laboratory, University of Maryland, 1986.

Kirkpatrick, S., Gelatt Jr., C.D. and Vecchi, M.P., "Optimization by simulated annealing", RC 9355 (#41093), IBM T.J. Watson Research Center, Yorktown Heights, NY.

Lee, D. and Pavlidis, T., personal communication, 1986.

Longuet-Higgins, H.C., "A computer algorithm for reconstructing a scene from two projections", *Nature*, **293**, 133–135, 1981.

Longuet-Higgins, H.C. and Prazdny, K., "The interpretation of a moving retinal image", *Proc. Royal Soc. London B*, **208**, 385–397, 1980.

Marr, D., *Vision*, W.H. Freeman, San Francisco, 1982.
Morozov, V.A., *Regularization Methods for Solving Ill-Posed Problems*, Springer-Verlag, 1984.
Nagel, H.H. and Neumann, B., "On 3-D reconstruction from two perspective views", *Proc. IJCAI*, 661–663, 1981.
Nalwa, V., "Detecting edges in images", personal communication, 1985.
Negahdaripour, S. and Horn, B.K.P., "Determining 3-D motion of planar objects from image brightness patterns", *Proc. IJCAI*, Los Angeles, CA, 898–901, 1985.
Poggio, T. and Koch, C., "Ill-posed problems in early vision: from computational theory to analog networks", *Proc. Royal Soc. London B*, **226**, 303–333, 1985.
Poggio, T. and the staff, "MIT progress in understanding images", *Proc. Image Understanding Workshop*, Miami, FL, 25–39, 1985.
Prazdny, K., "Determining the instantaneous direction of motion from optical flow generated by curvilinearly moving observer", *Computer Vision, Graphics and Image Processing*, **17**, 94–97, 1981.
Roach, J.W. and Aggarwal, J.K, "Determining the movement of objects from a sequence of images", *IEEE Transactions on PAMI*, **2**, 554–562, 1980.
Stevens, K., "The information content in texture gradients", *Biological Cybernetics*, **42**, 95–105, 1981.
Terzopoulos, D., "Regularization of inverse problems involving discontinuities", *IEEE Transactions on PAMI*, **8**, 413–425, 1986.
Tichonov, A.N. and Arsenin, V.Y., *Solutions of Ill-Posed Problems*, Winston, Washington, 1977.
Tsai, R.Y. and Huang, T.S., "Uniqueness and estimation of three dimensional motion parameters of rigid objects with curved surfaces", *IEEE Transactions on PAMI*, **6**, 13–27, 1984.
Ullman, S., "The interpretation of structure from motion", *Proc. Royal Soc. London B*, **203**, 405–426, 1979.
Waxman, A. and Ullman, S., "Surface structure and 3-D motion from image flow: a kinematic analysis", CAR-TR-24, Center for Automation Research, University of Maryland, October 1983.
Witkin, A., "Recovering surface orientation and shape from texture", *Artificial Intelligence*, **17**, 17–45, 1981.

APPENDIX 1

Theorem:

Let a coordinate system $OXYZ$ be fixed with respect to the left camera, with the Z axis pointing along the optical axis. We assume that the image plane Im_1 is perpendicular to the Z axis at the point (0,0,1) and that O is the nodal point of the left camera. Let the nodal point of the right camera be the point (R,L,O) and let its image plane be identical to the previous one, i.e. $\mathrm{Im}_1 = \mathrm{Im}_2$. Consider a polygon P on the world plane $Z = pX + qY + c$, defined by the points (X_i,Y_i,Z_i), $i = 1, \ldots, n$, and having area S_W. Let S_1, S_2 the areas of the paraperspective projections of P on the left and right cameras respectively and S_1', S_2' the areas of the perspective projections of the polygon P on the left and right cameras respectively. Then

$$\frac{S_1}{S_2} = \frac{S_1'}{S_2'}$$

Proof:

The proof is given in several parts.

Let (A_1,B_1) and (A_2,B_2) the centers of mass of the projections of the contour P on the left and right image planes respectively (it has to be noted that (A_1,B_1) and (A_2,B_2) are the centers of mass of the actual left and right images as opposed to the projections of the center of mass of P onto the left and right image planes). Then we have

$$\frac{S_2}{S_1} = \frac{1 - A_2 p - B_2 q}{1 - A_1 p - B_1 q} \tag{1}$$

The above equation is equation (6.17), which we will prove to be exact under perspective projection.

Now

$$A_1 = \frac{1}{n} \sum \left[\frac{X_i}{Z_i} \right], \quad B_1 = \frac{1}{n} \sum \left[\frac{Y_i}{Z_i} \right],$$

$$\text{and } A_2 = \frac{1}{n} \sum \left[\frac{X_i - R}{Z_i} \right], \quad B_2 = \frac{1}{n} \sum \left[\frac{Y_i - L}{Z_i} \right]$$

Substituting in (1) we get after some tedious manipulations

$$\frac{S_2}{S_1} = 1 + \frac{pR + qL}{c} \tag{2}$$

On the other hand, we can easily prove that

$$\frac{S_2}{S_1} = 1 + R \frac{\sum \left[\frac{Y_i - Y_{i+1}}{Z_i Z_{i+1}} \right]}{M} + L \frac{\sum \left[\frac{X_i - X_{i+1}}{Z_i Z_{i+1}} \right]}{M} \tag{3}$$

with

$$M = \sum \left[\frac{X_i Y_{i+1} - X_{i+1} Y_i}{Z_i Z_{i+1}} \right] \tag{4}$$

We can also easily prove that

$$\frac{\sum \left[\frac{Y_i - Y_{i+1}}{Z_i Z_{i+1}} \right]}{M} = \frac{p}{c} \tag{5}$$

and

$$\frac{\sum \left[\frac{X_i - X_{i+1}}{Z_i Z_{i+1}} \right]}{M} = \frac{q}{c} \tag{6}$$

From equations (2), (3), (4), (5) and (6) the proof of the theorem is immediate.

APPENDIX 2

Theorem:

With the nomenclature of the previous theorem (Appendix 1) and assuming only horizontal displacement of the cameras, if S_L, S_R denote the areas of the left and right images, and the world plane from which the image contour is obtained is $Z = pX + qY + c$, then

$$\frac{S_L}{S_R} = \frac{1}{1 + \frac{dx}{c} p}$$

(dx = displacement between camera 1 and camera 2).

Proof:

$$S_L = \frac{1}{2}\left(\sum x_i^L y_{i+1}^L - \sum x_{i+1}^L y_i^L\right) \quad \left[\begin{matrix}\text{Area of a}\\ \text{polygon}\end{matrix}\right]$$

$$S_R = \frac{1}{2}\left(\sum x_i^R y_{i+1}^R - \sum x_{i+1}^R y_i^R\right)$$

where (x^L, y^L), (x^R, y^R) are the coordinates in the left and right image planes, respectively.

Now $y_{i_8} = y_i^R$ (i)

and $x_i^R = \frac{f(x_i^L - dx)}{Z_i} = x_i^L - \frac{f\,dx}{Z_i}$ (ii)

So

$$\frac{S_L}{S_R} = \frac{\frac{1}{2}\left(\sum x_i^L y_{i+1}^L - \sum x_{i+1}^L y_i^L\right)}{\frac{1}{2}\left(\sum x_i^R y_{i+1}^R - \sum x_{i+1}^R y_i^R\right)}$$

$$= \frac{1}{\frac{\left(\sum\left(x_i^L - \frac{f\,dx}{Z_i}\right) y_{i+1}^L - \sum\left(x_{i+1}^L - \frac{f\,dx}{Z_{i+1}}\right) y_i^L\right)}{\left(\sum x_i^L y_{i+1}^L - \sum x_{i+1}^L y_i^L\right)}} \quad \left[\begin{matrix}\text{Using}\\ \text{(i) and (ii)}\end{matrix}\right]$$

$$= \frac{1}{1 + f\,dx\left(\frac{\sum \frac{y_{i+1}^L}{Z_i} - \sum \frac{y_i^L}{Z_{i+1}}}{\sum x_i^L y_{i+1}^L - \sum x_{i+1}^L y_i^L}\right)}$$

$$= \frac{1}{1 + \frac{dx}{c} p\left(\frac{fc}{p}\left(\frac{\sum \frac{y_i^L}{Z_{i+1}} - \sum \frac{y_{i+1}^L}{Z_i}}{\sum x_i^L y_{i+1}^L - \sum x_{i+1}^L y_i^L}\right)\right)}$$

Thus we need to show

$$\frac{fc}{p}\,\frac{\left(\sum \frac{Y_i^L}{Z_{i+1}} - \sum \frac{y_{i+1}}{Z_i}\right)}{\left(\sum x_i^L y_{i+1}^L - \sum x_{i+1}^L y_i^L\right)} = 1$$

i.e.

$$\sum y_i^L \frac{fc}{Z_{i+1}} - \sum y_{i+1}^L \frac{fc}{Z_i} = \sum (p\,x_i^L)\, y_{i+1}^L - \sum (p\,x_{i+1}^L)\, y_i^L$$

But

$$Z_i = p\,x_i^L + q\,y_i^L + c$$

$$\Rightarrow \frac{c}{Z_i} = 1 - p\frac{x_i^L}{Z_i} - 1\frac{y_i^L}{Z_i}$$

$$\Rightarrow \frac{fc}{Z_i} = f - p\frac{f\,x_i^L}{Z_i} - q\frac{f\,y_i^L}{Z} = f - p\,x_i^L - q\,y_i^L$$

Hence

$$\sum y_i^L \frac{fc}{Z_{i+1}} - \sum y_{i+1} \frac{fc}{Z_i}$$

$$= \sum y_i^L (f - p\,x_{i+1}^L - q\,y_{i+1}^L) - \sum y_{i+1}^L (f - p\,x_i^L - q\,y_i^L)$$

$$= f\left(\sum y_i^L - \sum y_{i+1}^L\right) + \left(\sum (p\,x_i^L)\, y_{i+1}^L - \sum (p\,x_{i+1}^L)\, y_{i_L}\right)$$

$$+ q\left(\sum y_i^L y_{i+1}^L - \sum y_{i+1}^L y_i^L\right) \quad \left[\begin{matrix}\text{Since, } y_{n+1} = y_1\\ \text{by notation}\end{matrix}\right]$$

$$= \sum (p\,x_i^L)\, y_{i+1}^L - \sum (p\,x_{i+1}^L)\, y_i^L$$

Thus we have proved the theorem.

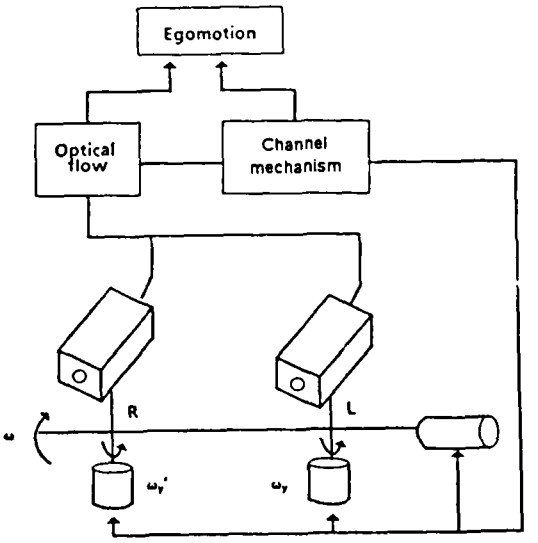
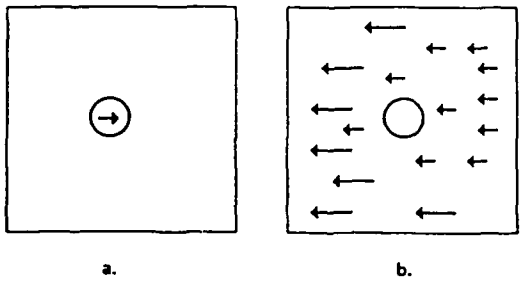
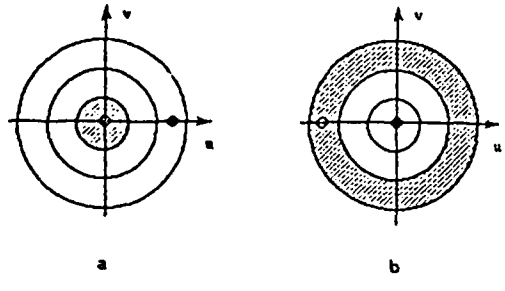


Figure 5.1 The Tracking Mechanism



The tracking system must use velocities that stem from the object being tracked and ignore background velocities. (a) Shows an initial situation where a target is moving on the retina. (b) Once the tracking system is engaged, the target is moving with a relative velocity near zero but the background has a large signal.

Figure 5.2 Target Identification



The global velocity space registers the number of flow vectors with certain values. Channels, shown in the figure as concentric annular regions, allow ranges of velocities to be selectively ignored. (a) Initially the low velocity channel is off (shaded) allowing the system to selectively register a moving target (•) and ignore background variations (○). (b) Once the tracking mechanism is activated the high velocity channels are blocked and again the target velocities are passed, ignoring the background signals.

Figure 5.3 Concept of the Velocity Channel

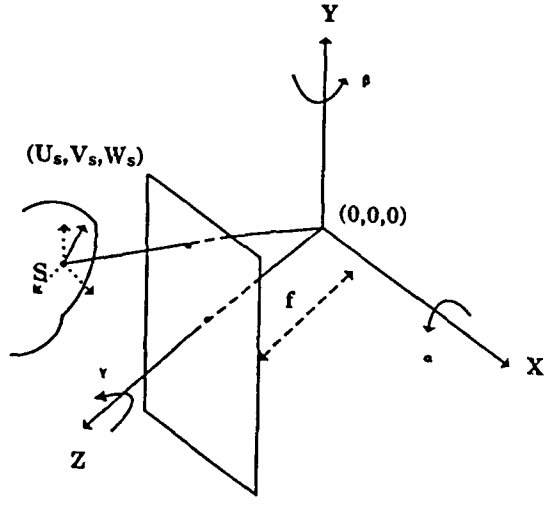


Figure 5.4 Imaging Geometry and Motion Representation

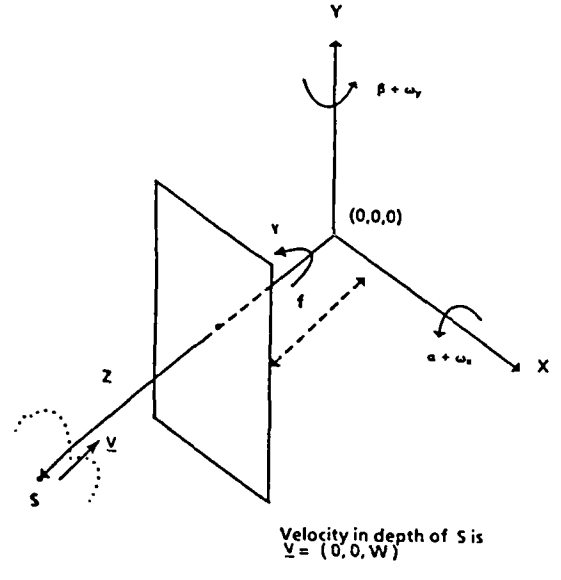


Figure 5.5 Monocular Tracking

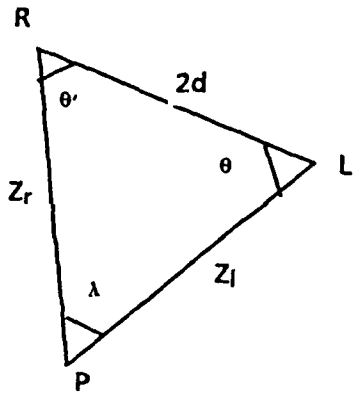


Figure 5.7 Binocular Convergent Tracking

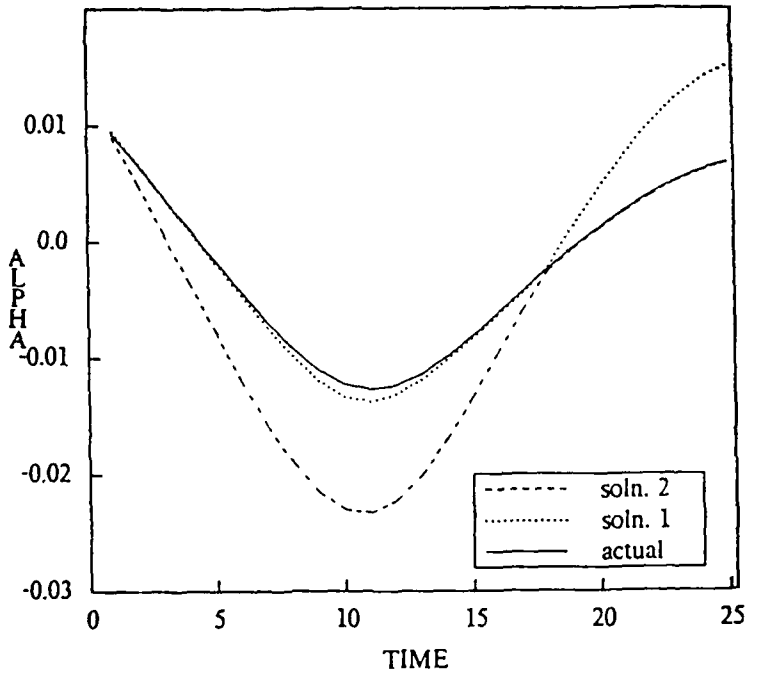


Figure 5.9 Time Evolution of Rotation about x-axis

AN INTEGRATED SYSTEM THAT UNIFIES MULTIPLE SHAPE FROM TEXTURE ALGORITHMS

Mark L. Moerdler and John R. Kender[1]

Department of Computer Science, Columbia University,New York, N.Y. 10027

ABSTRACT

This paper describes an approach which intelligently integrates several conflicting and corroborating shape-from-texture methods in a single system. The system uses a new data structure, *augmented texels*, which combines multiple constraints on orientation in a compact notation for a single surface patch. The augmented texels initially store weighted orientation constraints that are generated by the system's several independent shape-from-texture components. These texture components, which run autonomously and may run in parallel, derive constraints by any of the currently existing shape-from-texture approaches e.g. shape-from-uniform-texel-spacing. For each surface patch the *augmented texel* then combines the potentially inconsistent orientation data, using a hough transform-like method on a tesselated gaussian spheres, resulting in an estimate of the most likely orientation for the patch. The system then defines which patches are part of the same surface, simplifing surface reconstruction.

This knowledge fusion approach is illustrated by a system that integrates information from two different shape-from-texture methods, shape-from-uniform-texel-spacing and shape-from-uniform-texel-size. The system is demonstrated on camera images of artificial and natural textures.

1 INTRODUCTION

This paper proposes a new approach to the problem of defining and reconstructing surfaces based on multiple independent textual cues. The generality of this approach is due to the interaction between textural cues, allowing the methodology to extract shape information from a wider range of textured surfaces than any individual method. The method, as shown in figure 1, consists of three major phases, the calculation of orientation constraints and the generation of *texel patches*[2], the consolidation of constraints into a "most likely" orientation per patch, and finally the reconstruction of the surface.

During the first phase the different shape-from-texture components generate texel patches and *augmented texels*. Each augmented texel consists of the 2-D description of the texel patch and a list of weighted orientation constraints for the patch. The orientation constraints for each patch are potentially inconsistent or incorrect because the shape-from methods are locally based and utilize an unsegmented, noisy image.

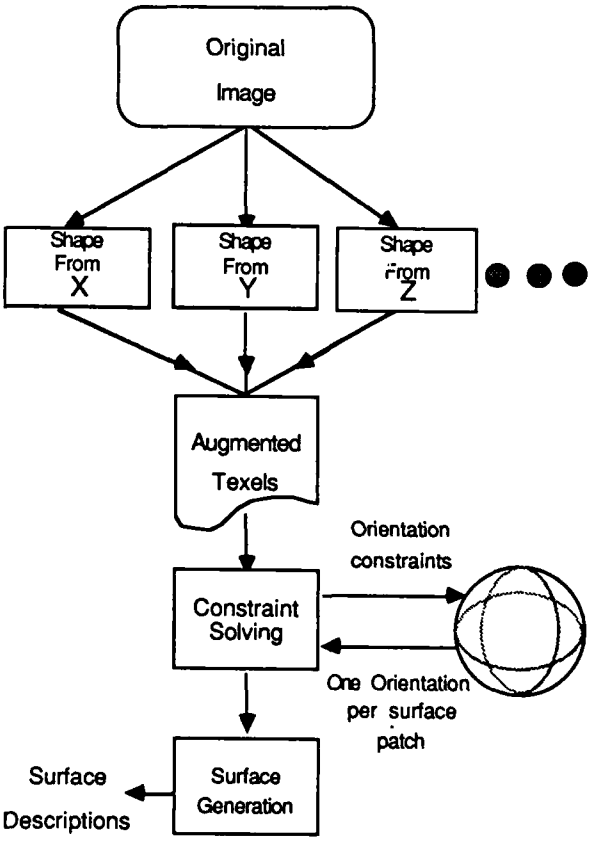


Figure 1: Integrating multiple shape-from methods

In the second phase, all the orientation constraints for each augmented texel are consolidated into a single "most likely" orientation by a Hough-like transformation on a tesselated Gaussian Sphere. During this phase the system will also merge together all augmented texels that cover the same area of the image. This is necessary because some of the shape-from components define "texel" similarly, and the constraints generated should also be merged.

Finally, the system re-analyzes the orientation constraints to

[1]This research was supported in part by ARPA grant #N00039-84-C-0165, by a NSF Presidential Young Investigator Award, and by Faculty Development Awards from AT&T, Ford Motor Co., and Digital Equipment Corporation.

[2]A *texel patch* is a 2-D description of a subimage that contains one or more textural elements. The number of elements that compose a patch is dependent on the shape-from-texture algorithm.

determine which augmented texels are part of the same constraint family and groups them together. In effect, this segments the image into regions of similar orientation. The surface can then be reconstructed from these surface patches.

The robustness of this approach is illustrated by a system that fuses the orientation constraints of two existing shape-from methods: shape-from-uniform-texel-spacing [Moerdler and Kender 85], and shape-from-uniform-texel-size [Ohta et. al. 81]. These two methods generate orientation constraints for different overlapping classes of textures.

2 HISTORICAL BACKGROUND

Current methods to derive shape-from-texture are based on measuring a distortion[3] that occurs when a textured surface is viewed under perspective. This perspective distortion is imaged as a change in some aspect of the texture. In order to simplify the recovery of the orientation parameters from this distortion, researchers have imposed limitations on the applicable class of textured surfaces. Some of the limiting assumptions include uniform texel spacing [Kender 80; Kender 83; Moerdler and Kender 85], uniform texel size [Ikeuchi 80; Ohta et. al. 81; Aloimonos and Swain 85], uniform texel density [Aloimonos 86], and texel isotropy [Witkin 80; Davis, Janos and Dunn 83; Dunn 84]. Each of these are strong limitations causing methods based on them to be appliable to only a limited range of real images.

3 DESIGN METHODOLOGY

The generation of orientation constraints from perspective distortion uses one or more image texels. The orientation constraints can be considered as local, defining the orientation of individual surface patches (called *texel patches*[4]) each of which covers a texel or group of texels. This definition allows a simple extension to the existing shape-from methods beyond their current limitation of planer surfaces or simple non planer surfaces based on a single textural cue. The problem can then be considered as one of intelligently fusing the orientation constraints per patch. Ikeuchi [Ikeuchi 80] and Aloimonos [Aloimonos and Swain 85] attempt a similar extension based on constraint propagation and relaxation for planer and non planer surfaces for using only a single shape-from-texture method.

The process of fusing orientation constraints and generating surfaces can be broken down into the following three phases:

1. The creation of texel patches and multiple orientation constraints for each patch.
2. The unification of the orientation constraints per patch into a "most likely" orientation.
3. The formation of surfaces from the texel patches.

Each of the remaining subsections of this chapter describes one of these phases.

[3]Under the assumption that natural texture does not mimic projective effects nor does it cancel those effects out.

[4]Texel patches are defined by how each method utilizes the texels. Some methods (e.g. Uniform texel size) use a measured change between two texels; in this case the texels patches are the texels themselves. Other methods (e.g. Uniform texel density) use a change between two areas of the image, in this case the texel patches are these predefined areas.

3.1 SURFACE PATCH AND ORIENTATION CONSTRAINT GENERATION

The first phase of the system consists of multiple shape-from-texture components which generate augmented texels. Each augmented texel consisting of a texel patch, orientation constraints for the texel patch, and an assurity weighting per constraint. The orientation constraints are stored in the augmented texel as vanishing points which are mathematically equivalent to a class of other orientation notations (e.g. Tilt and Pan constraints) [Shafer, Kanade and Kender 83]. Moreover, they are simple to generate and compact to store.

The assurity weighting is defined separately for each shape-from method and is based upon the intrinsic error of the method. For example, shape-from-uniform-texel-spacing's assurity weighting is a function of the total distance between the texel patches used to generate that constraint. A low assurity value is given when the inter-texel distance is small (~3 pixels) because under these conditions a small digitization error causes a large orientation error. As the inter-texel distance increases the assurity value also increases. At a heuristic threshold, it starts to decrease again based on the fact that once the inter texel distance grows too large the local surface is no longer approximated by a plane and the orientation error grows.

3.2 MOST LIKELY ORIENTATION GENERATION

Once the orientation constraints have been generated for each augmented texel, the next step consists of unifying the constraints into one orientation per augmented texel. The major difficulty in deriving this "most likely" orientation is that the constraints are errorful, inconsistent, and potentially incorrect. A simple and computationally feasible, solution to this is to use a Gaussian Sphere which maps the orientation constraints to points on the sphere [Shafer, Kanade and Kender 83]. A single vanishing point circumscribes a great circle on the Gaussian Sphere; two different constraints generate two great circles that overlap at two points uniquely defining the orientation of both the visible and invisible sides of the surface patch.

The Gaussian sphere is approximated, within the system, by the hierarchical by tesselated Gaussian Sphere based on trixels (triangular shaped faces. See figure 2) [Ballard and Brown 82; Fekete and Davis 84; Korn and Dyer 86]. The top level of the hierarchy is the icosahedron. At each level, other than the lowest level of the hierarchy, each trixel has four children. This hierarchical methodology allows the user to specify the accuracy to which the orientation can be calculated by defining the number of levels of tesselation that are created.

The system generates the "most likely" orientation for each texel patch by accumulating evidence for all the constraints for the patch. For each constraint, it recursively visits each trixel to check if the constraint's great circle falls on the trixel, and then visiting the children if the result is positive. At each leaf trixel the likelihood value of the trixel is incremented by the constraint's weight. The hierarchical nature of this approach limits the number of trixels that need to be visited.

Once all of the constraints for a texel patch have been considered, a peak finding program smears the likelihood values at the leaves. Currently, this is done heuristically: the smeared value of each leaf is equal to 1/2 the value of its neighbors plus 1/4 the value of all its neighbor's neighbors that are not a neighbor of leaf. This is a rough approximation to a gaussian blur. The "most likely" orientation is defined to be the trixel with the largest smeared value.

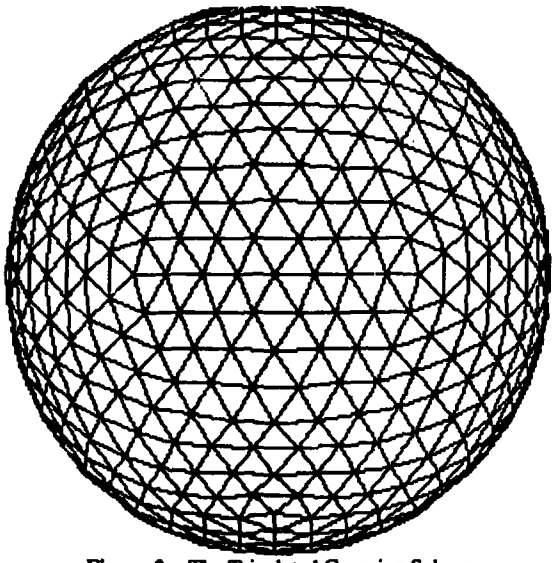


Figure 2: The Trixelated Gaussian Sphere

3.3 SURFACE GENERATION

The final phase of the system generates surfaces from the individual augmented texels. This is done by re-analyzing the orientation constraints generated by the shape-from methods in order to determine which augmented texels are part of the same surface. In doing this the surface generation is also performing a first approximation of surface separation and segmentation.

The re-analysis consists of iterating through each Augmented Texel, considering all its orientation constraints and determining which constraints aided in defining the "correct" orientation for the texel patch as described in phase two. If an orientation constraint correctly determined the orientation of all the texels that were used in generating the constraint then these augmented texels are considered as part of the same surface.

Once it is determined which augmented texels are part of the same surface, the surfaces are generated by a simple planer surface generation algorithm. The surfaces are defined as the best fit approximation that contains all the related texel patches. This approach allows both the generation and the separation of surfaces that are connected or overlapping.

4 TEST DOMAIN

The knowledge fusion approach outlined in the previous section has been applied to a test system that contains two shape-from-texture methods, shape-from-uniform-texel-spacing [Moerdler and Kender 85], and shape-from-uniform-texel-size [Ohta et. al. 81]. Each of the methods is based on la different, limited type of textures to which they are applicable. Shape-form-uniform-texel-spacing derives orientation constraints based on the assumption that the texels on the surface are of arbitrary shape but are equally spaced, while shape-from-uniform-texel-size is based on the unrelated criteria that the spacing between texels can be arbitrary but the size of all of the texels are equivalent but unknown.

In shape-from-uniform-texel-size if the distance from the center of mass of texel T_1 to texel T_2 (see figure 3) is defined as D then the distance from the center of texel T_2 to a point on the vanishing line can be rewritten as :

$$F_2 = D \times S_2^{1/3} / (S_1^{1/3} - S_2^{1/3})$$

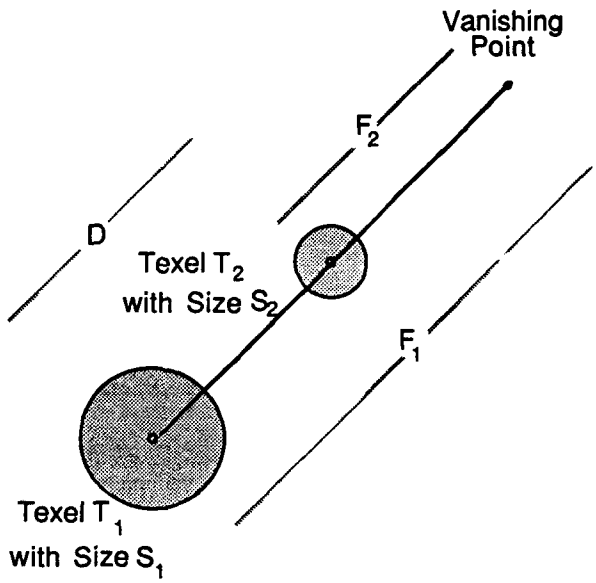


Figure 3: The calculation of shape-from-uniform-texel-size

In shape-from-uniform-texel-spacing the calculations are similar. Given any two texels T_1 and T_2 (see figure 4) whose relative positions are P_1 and P_2, if the distance from T_1 to the mid-texel T_3 is equal to L and the distance from T_2 to the same mid-texel T_3 is equal to R, the vanishing point distance X is given by :

$$X=[L\times P_1 - R\times P_2] / [L-R]$$

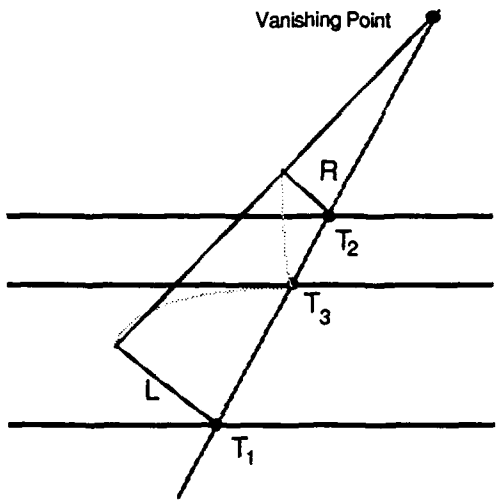


Figure 4: A geometrical representation of back-projecting.

Under certain conditions either method may generate incorrect constraints which are ignored. On textures that are solvable by both methods the methods cooperate and correctly define the textured surface or surfaces in the image. Some images are not solvable by

either method by itself but can only be correctly segmented and the surfaces defined by the interaction of the cues (i.e. the upper right texel of figure 15).

5 THE EFFECTS OF NOISE

The real, camera generated, images contains noise and shadows which are effectly ignored by the system in many cases. The system treats shadows as potential surface texels (see texels 9 and 13 in figure 5) and uses them to compute orientation constraints. Since many texels are used in generating the orientation for each individual texel the effect of shadow texels is minimized[5].

Noise can occur in many ways: it can create texels, and it can change the shape, size, or position of texels. If noise texels are sufficiently small then they are ignored in the texel finding components of the shape-from methods. When they are large, they are treated in much the same way as shadow texels and thus often do not effect the orientation of the surface texel patches. Since many texels are used and more than one shape-from method is employed, noise-created changes in the shape of texels can perturb the orientation results, but the effect appears negligible as shown in the experimental results.

6 EXPERIMENTAL RESULTS

The system has been tested over a range of both synthetic and natural textured surfaces, and appears to show robustness and generality. Three examples are given on real, noisy images that demonstrate the cooperation among the shape-from methods.

The first image, figure 5, shows a real image of a man-made texture consisting of equally spaced, equally sized circles. The system finds fifteen texels: the twelve texels on the surface, plus three noise texels located in the background. It is able to generate the correct q value for each of the twelve surface texels and a p value that is off by 2 degrees (see figure 7 for the positions of the texels and figure 9 for the individual p and q values.) In figure 8 the orientations of the texel patches are displayed as needle-like surface normal vectors.

The system is also also able to segmented the image into surfaces, one of which contains only the twelve correct surface texels. The noise generated texels are each individually marked as part of separate surfaces.

The second example consists of a surface textured with twelve uniformly sized coins (see figure 10). The system in this case generates p and q values that are approximately correct (two degrees for the p and exactly correct for the q. See figure 13) for ten of the twelve texel patches. The other two texels have a greater error (five degrees for the p and correct for the q) due to digitization errors (see figure 14). The constraints generated by the shape-from-uniform-texel-spacing algorithm are ignored.

The final example is an image of a box of breakfast buns in which one bun is missing (see figure 15). Eleven texels are found, eight of which are patches of filling on the surface of buns; two are noisy data on the buns; and one is part of the packaging. The texels are only approximately evenly spaced and approximately evenly

[5]Even under the conditions where many shadow texels are found they do not effect the computed orientation of surface texels so long as the placement of the shadow texels does not mimic perspective distortion.

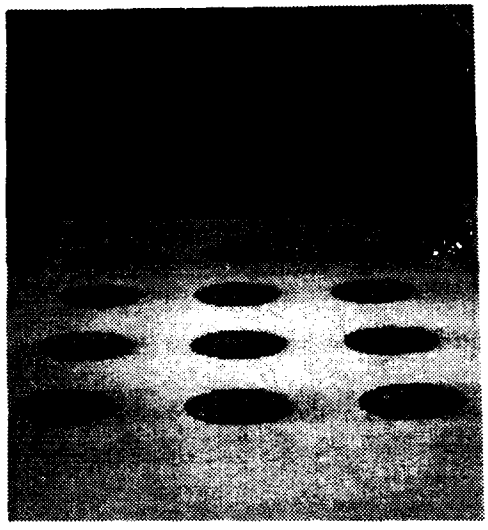


Figure 5: Surface textured with evenly spaced evenly sized circles sized.

The system is able to generate approximately correct surface (see figure 16) normals for all but one of the texels on the breakfast buns and an approximate orientation for the outside of the package. The upper left hand texel contains two equally weighted orientations. Without the shape-from-uniform-texel-spacing method the correct orientation would not have been generated for this patch.

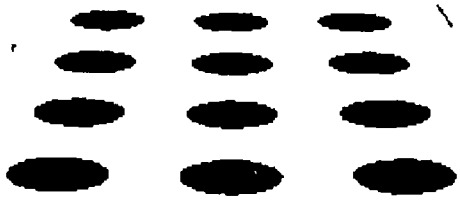


Figure 6: The texels found for the evenly spaced circles texture

7 CONCLUSION AND FUTURE RESEARCH

In this paper a system has been proposed that can fuse the results of a number of shape-from-texture methodologies to generate surface segments and their orientations. The system has been tested using two existing shape-from methods, shape-from-uniform-texel-spacing and shape-from-uniform-texel-size and has shown the ability to recover the surfaces and their orientation under noisy conditions.

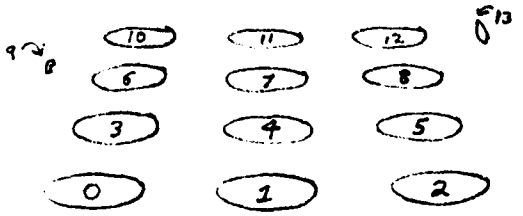


Figure 7: The numbering of texels for the circles texture

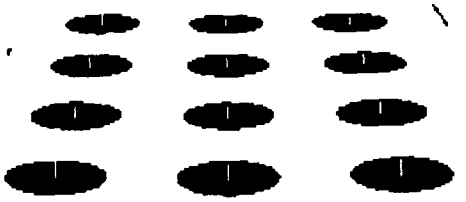


Figure 8: Surface normals for the surface textured with circles

Texel Number	Measured p & q	Actual p & q	% error
0 to 8	p =2.6 q =0.0	p = 3.0 q = 0.0	2^o 0^o
9	p=11.0 q=6.7	Shadow Texel	
10 to 12	p =2.6 q =0.0	p = 3.0 q = 0.0	2^o 0^o
13	p=11.0 q=6.7	Shadow Texel	

Figure 9: Orientation values for the surface textured with circles

The robustness of the system has been exercised using images that contain multiple surfaces, by surfaces that are solvable by either method alone, and finally by surfaces that are solvable by using only both methods together.

Future enhancements to the system would include the addition of other shape-from-texture modules, the investigation of other means of fusing information (such as object model approaches), and optimization of the method, especially in a parallel processing environment.

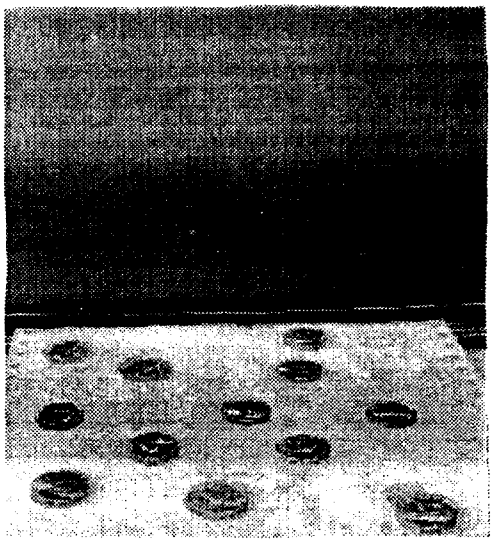


Figure 10: An image of a surface covered with coins

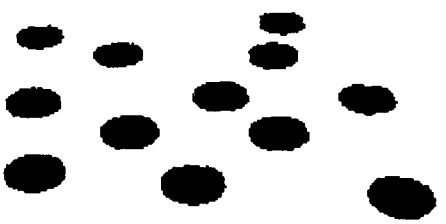


Figure 11: The coins found in the image

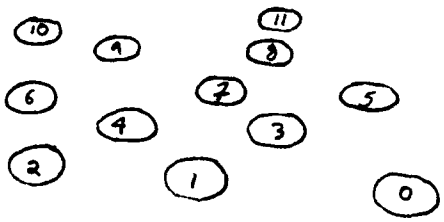


Figure 12: The texel numbers of the coins

Texel Number	Measured p & q	Actual p & q	% error
0	p=5.5 q=0.0	p = 3.0 q = 0.0	5^o 0^o
1 to 4	p =2.6 q =0.0	p = 3.0 q = 0.0	2^o 0^o
5	p=5.5 q=0.0	p = 3.0 q = 0.0	5^o 0^o
6 to 9	p =2.6 q =0.0	p = 3.0 q = 0.0	2^o 0^o
10 & 11	p=5.5 q=0.0	p = 3.0 q = 0.0	5^o 0^o

Figure 13: Orientation values for the surface textured with coins

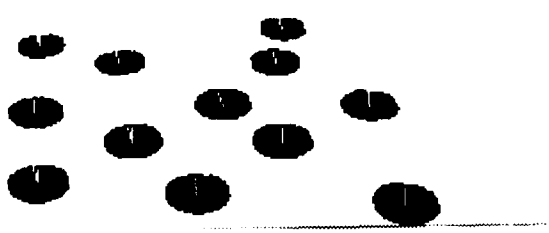


Figure 14: surface normals generated for the coins

Figure 15: A box of breakfast buns with one bun missing

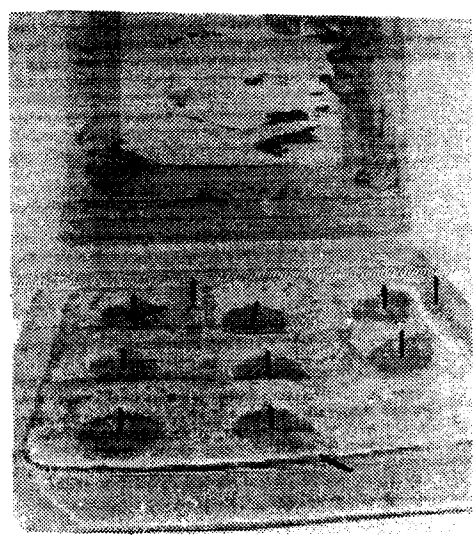


Figure 16: The surface normals generated for the box of buns

References

[Aloimonos 86] John Aloimonos.
Detection of Surface Orientation and Motion from Texture: 1. The Case of Planes.
In *Proceedings of Computer Vision Pattern Recognition Conference*. Computer Vision Patttern Recognition, IEEE Computer Society, 1986.

[Aloimonos and Swain 85]
John Aloimonos and Michael J. Swain.
Shape from Texture.
In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*. IJCAI, IJCAI, 1985.

[Ballard and Brown 82]
Dana Ballard and Christopher Brown.
Computer Vision.
Prentice-Hall Inc., 1982.

[Davis, Janos and Dunn 83]
Larry S. Davis, Lubvik Janos, and Stanley M. Dunn.
Efficient Recovery of Shape from Texture.
IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-5(5):485-492, September 1983.

[Dunn 84] Stanley M. Dunn, Larry S. Davis, and Hannu A. Hakalahti.
Experiments in Recovering Surface Orientation from Texture.
Technical Report CAR-TR-61, University of Maryland Center For Automation Research, May 1984.

[Fekete and Davis 84]
Gyorgy Fekete and Larry S. Davis.
Property Spheres: A New Representation For 3-D Object Recognition.
Proceedings of the Workshop on Computer Vision Representation and Control :192 - 201, 1984.

[Gibson 50] James J. Gibson.
Perception of the Visual World.
Riverside Press, 1950.

[Ikeuchi 80] Katsushi Ikeuchi.
Shape from Regular Patterns (an Example from Constraint Propagation in Vision).
Proceedings of the Internation Conference on Pattern Recognition :1032-1039, December 1980.

[Kanatani and Chou 86]
Ken-ichi Kanatani and Tsai-Chia Chou.
Shape from Texture: General Principle.
In *Proceedings of Computer Vision Pattern Recognition Conference*. Computer Vision Patttern Recognition, IEEE Computer Society, 1986.

[Kender 80] John R. Kender.
Shape from Texture.
PhD thesis, C.M.U., 1980.

[Kender 83] John R. Kender.
Surface Constraints from Linear Extents.
Proceedings of the National Conference on Artificial Intelligence , March 1983.

[Korn and Dyer 86]
Matthew R. Korn and Charles R. Dyer.
3-D Multiview Object Representation for Model-Based Object Recognition.
Technical Report RC 11760, IBM T.J. Watson Research Center, March 86.

[Moerdler and Kender 85]
Mark L. Moerdler and John R. Kender.
Surface Orientation and Segmentation from Perspective Views of Parallel-Line Textures.
Technical Report, Columbia University, 1985.

[Ohta et. al. 81] Yu-ichi Ohta, Kiyoshi Maenobu, and Toshiyuki Sakai.
Obtaining Suface Orientation from Texels under Perspective Projection.
In *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*. IJCAI, IJCAI, 1981.

[Shafer, Kanade and Kender 83]
Steven A. Shafer and Takeo Kanade and John R. Kender.
Gradient Space under Orthography and Perspective.
Computer Vision, Graphics and Image Processing (24), 1983.

[Stevens 79] Kent A. Stevens.
Surface Perception from Local Analysis of Texture and Contour.
PhD thesis, Massachusetts Institute of Technology, 1979.

[Tsai 86] Roger Y. Tsai.
An Efficient and Accurate Camera Calibration Technique for 3D Machine Vision.
In Proceedings of Computer Vision Pattern Recognition Conference. Computer Vision Patttern Recognition, IEEE Computer Society, 1986.

[Witkin 80] Andrew P. Witkin.
Recovering Surface Shape from Orientation and Texture.
In Michael Brady (editors), *Computer Vision*, pages 17-45. North-Holland Publishing Company, 1980.

DRIVE - Dynamic Reasoning from Integrated Visual Evidence

Bir Bhanu and Wilhelm Burger

Honeywell Systems & Research Center
3660 Technology Drive, Minneapolis, MN 55418

ABSTRACT

The vision system of an Autonomous Land Vehicle is required to handle complex dynamic scenes. Vehicle motion and individually moving objects in the field of view contribute to a continuously changing camera image. The goal of "motion understanding" is to find consistent three-dimensional interpretations for those changes in the image sequence. The estimation of vehicle motion, the detection of moving objects in the scene and the description of their movements are the key issues. We present a new approach to this problem, which departs from previous work by emphasizing a qualitative line of reasoning and modeling, where multiple interpretations of the scene are pursued simultaneously. Different sources of visual information are integrated in a rule-based framework to construct and maintain a vehicle-centered three-dimensional model of the scene. It is shown that this approach offers significant advantages over "hard" numerical techniques which have been proposed in the motion understanding literature.

1. INTRODUCTION

Visual information from the environment is an indispensable clue for the operation of the Autonomous Land Vehicle (ALV). Even with the use of sophisticated inertial navigation systems, the accumulation of position errors requires periodic corrections. Mission tasks involving search and rescue, exloration and manipulation (e.g. refueling and munitions deployment) critically depend on visual information. Motion becomes a natural component as soon as moving objects are encountered in some form, while following a convoy, approaching other vehicles or detecting threats. The presence of moving objects and their behavior must be known to provide appropriate counteraction. In addition to that, image motion provides important cues about the spatial layout of the environment and about the actual movements of the vehicle. As part of the vehicle control loop, visual motion feedback is beneficial for path stabilization, steering and braking. Results from psychophysics[9,16] show, that humans rely heavily on visual motion for motor control.

While the vehicle is moving itself, the entire camera image is changing continuously. The interpretation of complex dynamic scenes is therefore the continuous task for the vision system of an autonomous land vehicle. Any change in the 2-D image is always the result of a change in 3-D space, either induced by vehicle motion or by moving objects. Finding a consistent interpretation for every change in the image is the objective of "motion understanding".

Previous work in motion analysis has mainly concentrated on numerical approaches for reconstructing motion and scene structure from image sequences. Recently Nagel[13] has given a comprehensive review. While a completely stationary environment has been assumed in most previous work on the reconstruction of camera motion, the possible presence of moving objects must be accounted for in this scenario. Similarly, we cannot rely on a fixed camera setup to detect those moving objects. Clearly, some kind of common reference is required, against which the movement of the vehicle as well as the movement of objects in the scene can be related.

Extensive work has been done in determining the relative motion and rigid 3-D structure of a given set of image points, basically following two approaches. In the first approach, structure and motion are computed in one integral step by solving a system of linear or nonlinear equations[11,12,19,23] from a minimum number of points on a rigid object. The method is reportedly sensitive to noise.[4,22] Recent work[2,3,6,18,21] has addressed the problem of recovering and refining 3-D structure from motion over extended periods of time, demonstrating that fairly robust results can be obtained. However, these approaches require distinct views of the object (the environment), which are generally not available to a moving vehicle. Besides that it seems, that the noise problem cannot be overcome by simply increasing the time of observation.

The second approach[2,7,9,10,14,17] makes use of the unique expansion pattern, which is experienced by a moving observer. Arbitrary observer motion can be decomposed into translational and rotational components from the 2-D image, without computing the structure of the scene. Figure 1 shows the basic viewing geometry for a moving observer. The vector representing the direction of instantaneous heading intersects the image plane at the "focus of expansion" (FOE). In a stationary environment every point in the image seems to expand from the FOE. The closer a point is in 3-D, the more rapidly it expands in the image. Thus for a stationary scene, its three-dimensional structure can be obtained

from the expansion pattern. Some obvious cases of object motion are easy to detect, such as motion towards the FOE, whereas other types of motion are not so obvious.

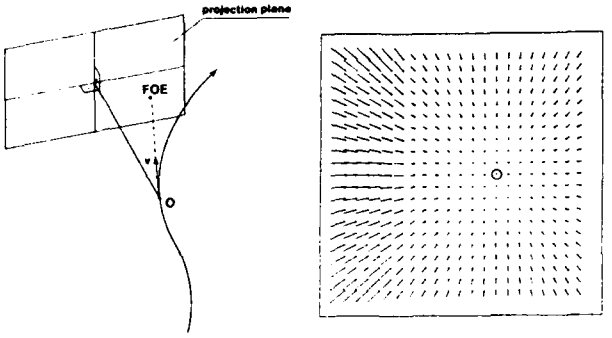


Figure 1. The basic viewing geometry for a moving observer[15] (*left*). The Focus of Expansion points at the direction of instantaneous heading. Stationary points in the scene seem to *expand* from one point (the FOE) when the camera moves along a straight line[17] (*right*). The rate of expansion indicates how close to the projection plane a point is in 3-D.

Our approach builds on the second method, using the expanding field of displacement vectors as the main source of information. Following a generate-and-test strategy, hypotheses about the structure of the environment are created from relationships in the image, which are checked for inconsistencies over time. Instead of creating a numerical description, the interpretation of the scene is accomplished in a qualitative fashion.[8]

The approach we are presenting is unique in several respects. First, a qualitative line of reasoning and modeling is pursued, which offers significant advantages over "hard" numerical techniques. Second, motion understanding is attempted in a hypothesize-and-test environment, where multiple interpretations of the scene may be active at the same time. In addition to the two-dimensional displacement field, we employ multiple sources of information, such as spatial reasoning and semantics for the construction and maintenance of the scene model.

The choice of a suitable spatial description of the environment[5] is an important component of motion understanding. A vehicle-centered model of the scene is constructed and maintained over time, representing the current set of feasible interpretations of the scene. In contrast to most previous approaches, we do not attempt an accurate geometric description of the scene. Instead we propose a *Qualitative Scene Model*, which holds only a coarse qualitative representation of the three-dimensional environment. As part of this model, the "stationary world" is represented by a set of image locations, forming a rigid 3-D configuration which is believed to be stationary. All the motion-related processes at the intermediate level use this model as a central reference. The vehicle's self-motion, for instance, must be related to the stationary parts of the environment, even if large parts of the image are in motion. We argue that this kind of qualitative modeling and reasoning is sufficient and efficient for the problem at hand.

While most previous work can be related to low-level computer vision, the integration of motion into higher levels, i.e. actual motion *understanding* is still in its infancy. Here we present a set of motion-related processes allocated at an *intermediate level* of computer vision. These processes bridge the representational gap between purely two-dimensional, image-centered low-level processes and the three-dimensional, world-centered high-level processes. High-level processes keep a global view of the scene, characterized by goal-directed activities (focus of attention). The long-term interactions between objects are handled at this level. The intermediate level performs "routine" tasks, which do not require focused attention, but short-term memory and knowledge. As far as motion is concerned, we want a vision system to be "surprised" only in cases of really *unexpectable* changes in the image. Changes due to normal self-motion, short-term occlusion etc. should not invoke attentative action at the high level.

2. UNDERSTANDING MOTION

PROBLEM STATEMENT

The purpose of our approach is to construct and maintain a consistent interpretation of time-varying images obtained from the camera on a moving vehicle. Specifically we are interested to determine

- how is the ALV moving ?
- what is moving in the scene and how does it move ?
- what is the approximate 3-D structure of the scene ?

The original input is a sequence of images, taken at a constant rate. A low-level correspondence technique is available, which extracts distinct features (points, lines or regions) from every image and supplies the 2-D displacement between successive frames for every feature. This is the actual input to our vision process. Recently a promising technique[1] for estimating the displacement field around region boundaries was devised, which makes this class of features attractive for our approach. At the present stage *points* are the only type of features being used. We assume that the problem of computing reliable displacement vectors for individual points (the "Correspondence Problem")[20] is solved. The correspondence algorithm labels each feature and tracks it over time by generating a list of tupels:

$$(\textit{feature-label}, t_0, x_0, y_0)$$
$$(\textit{feature-label}, t_0+1, x_1, y_1)$$
$$\ldots$$
$$(\textit{feature-label}, t_0+k, x_k, y_k)$$

When no correspondence is found, either because of occlusion or because the feature left the field of view, the list of tupels for this feature is discontinued. The low-level process is not expected to track a feature through occlusion, this is the task of intermediate-level processes, where short-term memory is available.

DETERMINATION OF VEHICLE MOTION

The first step of our approach is to determine the vehicle's motion relative to the stationary environment between each pair of frames. If the vehicle moves along a straight line, all stationary features seem to expand from one single point in the image, the focus of expansion. In the general case, vehicle rotation induces an additional vector field in the image. The fact that all the displacement vectors of stationary features must intersect in a common point can be used to "derotate" the image and compute the vehicle's rotation and direction of translation. Due to inertia, the direction of translation as well as the amount of rotation about either axis cannot change drastically between two frames. Thus the previous motion parameters can serve as a good guess for finding the current optimum.

We do not assume that the exact location of the FOE in the image plane can always be determined, but that a region can be specified, which contains the FOE with high certainty. However, the smaller the region of possible FOE-locations is, the more conclusions can be drawn for the interpretation of the scene. Although this first step is done exclusively in the image plane, knowledge about the stationary features to be used must be available in some form. This information is supplied by the *Qualitative Scene Model*, which is described in the following section.

Given the accurate location of the FOE (x_e,y_e), the relative range of any (stationary) point P in the image can be determined at time t by the relation

$$Z(t) \propto V(t)\,\frac{r(t)}{v(t)}\,,$$

where $Z(t)$ is the actual distance of the point from the image plane in 3-D, $V(t)$ is the velocity dZ/dt of the vehicle perpendicular to the image plane. $r(t)$ is the 2-D distance between the image of P and the FOE, $v(t)$ is the radial velocity dr/dt. Since $V(t)$ is the same for any stationary point in the scene, $r(t)$ and $v(t)$ can be measured in the image and depth can be reconstructed up to a common scale factor.

While the vehicle is traversing the environment, $Z(t)$ keeps changing for every feature in the field of view. If the depth map itself was used as the scene model, the model would have to be updated continuously, since the depth of objects is changing as the ALV moves forward. The topology of a stationary scene, however, remains unchanged. The key idea of our approach is to construct and maintain a topological model of the scene from observations in the time-varying images. If, as an example, for two 3-D features A,B with image points a,b

$$\frac{r_a(t)}{v_a(t)} < \frac{r_b(t)}{v_b(t)}$$

for every possible location of the FOE, then we may conclude that A is actually *closer* to the image plane than B. If A and B are really stationary, this relationship must hold in any future observation. Otherwise it can be concluded that at least one of the two features is *not* stationary.

DETECTING 3-D MOTION

A central issue of motion understanding is the detection of actual movements in the 3-D environment. Given the location of the FOE and the derotated displacement field (as explained earlier), some forms of 3-D motion are easy to detect, whereas others are of a more subtle nature. For instance, motion of an image point *towards* the FOE is a striking evidence of 3-D motion in the scene. In this particular case, something is obviously moving into the ALV's trajectory. This is, however, not the most "dangerous" case. Consider the situation in Figure 2, seen from the ALV while it is approaching an intersection. The shaded area in the center represents the region of possible locations of the FOE. The building on the right seems to expand away from the FOE, while the truck (which is actually moving) stays at a constant image location.

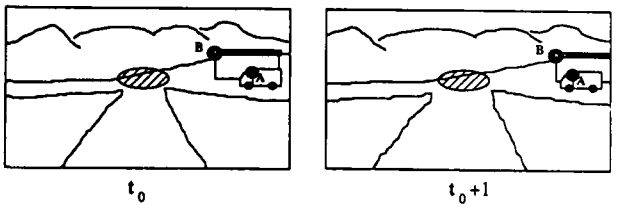


Figure 2. A typical situation: the ALV is approaching an intersection. Two feature points are tracked in the image, point A on the (moving) truck, and point B on the building. The stationary part of the image seems to *expand*, while the truck stays at a constant image location. The shaded area in the center represents the region of possible FOE-locations.

From the expansion pattern alone, the feature point A on the truck will be interpreted as *stationary* at *infinite* distance. However, if the truck and the ALV keep moving in this manner (with the image of the truck staying at the same location), the two will inevitably collide. Although in reality the truck is unlikely to show no image motion at all, the detection of this threat in the presence of noise is not trivial. It is only possible, when changes in the image can be related to the spatial layout of the scene. This allows reasoning in a fashion like:

> "Since the truck is occluding the building, it must be closer to the ALV than the building. The image of the building is expanding, thus it is at a finite distance. Therefore the truck cannot be at infinite distance (the original interpretation was wrong). Probably the truck is moving towards the ALV."

From knowledge about the *spatial* relationships between features in 3-D and their motion in the image, we can make conclusions about actual motion in the 3-D environment. However, different sources of knowledge and different paths of reasoning must be combined to come up with unambiguous interpretations.

OCCLUSION ANALYSIS

Whenever new image features appear in the field of view, they must be categorized and integrated into the current interpretation(s) of the scene. Thus the appearance of a feature constitutes an event of major activity for a motion analysis system. The same can be said for the disappearance of an image feature. The appearance of a feature can happen, when

- the feature had not been in the field-of-view before,
- the feature had been occluded, or
- the feature is moving.

Every feature must be taken care of at least once, when it becomes visible for the first time. If no global "world model" is available, this class of events cannot be predicted by any means. In contrast, given an appropriate model of the scene, the disappearance of a feature might well be predicted. This is not only important for understanding the event of disappearance itself, but it provides additional information about the consistency of the model. If, for instance, a stationary feature *A* does not occlude another feature *B* within the predicted period of time, either the spatial relationships assumed by the model are incorrect, or one of the two features is moving. In either case the model requires an update.

If the ALV is moving in an environment containing a large number objects at different depths, *temporary* occlusion of features will happen frequently. Whenever a feature becomes invisible, the low-level *correspondence process* will stop tracking this feature and discontinue the associated list of position-tupels (see above). This happens even if the feature is invisible for only one frame period, since the low-level process does not estimate the image-trajectory of a feature. When the feature finally reappears out of a temporary occlusion, the low-level process assigns a new *feature-label* to it and resumes to track it in the image just like a feature which has never been encountered before. The intermediate-level motion process, however, has most of the information available to *predict* the reappearance of a feature from temporary occlusion. This not only reduces the amount of "surprise" which must be handled in such an event, but can also *direct* the low-level process to look out for a new feature in a certain image region. Since this is part of a larger project for ALV-related vision the integration of map information for occlusion analysis and tracking is among the final goals.

RULES

There are various independent sources that may contribute to the creation of scene interpretations: *geometry*, *spatial reasoning*, and *semantics*. Each of these three types of knowledge is cast into a set of rules, which is applied to time-varying images.

Conclusions from the expansion pattern in the described above are based only on the *geometry* of the imaging process and the vehicle motion. The following set of geometric rules should illustrate the idea.

Rule *GEOMETRY-1*: This simple rule can be applied after the image has been derotated and the region of possible FOE locations has been determined.

if
point a is moving toward the FOE region
then
hypothesize (mobile a).

Rule *GEOMETRY-2*: If the image has been derotated and the region of possible FOE locations is known, then for every point a the minimum and maximum distance r_a^{min}, r_a^{max} and velocity v_a^{min}, v_a^{max} relative to the FOE can be computed.

if
(*stationary* a)
(*stationary* b)
($r_a^{max}/v_a^{min} < r_b^{min}/v_b^{max}$)
then
hypothesize (closer a b).

Since the relative distance between image points is not affected by small amounts of vehicle rotation, the image needs not to be derotated in the following rule. Only a very coarse estimate of the FOE's location must be available, which may be obtained from previous frames.

Rule *GEOMETRY-3*:

An image point a is said to be *inside* a point b, if a is closer to the FOE than b. For pairs of points in a certain neighborhood, this is easy to determine. $D_{a,b}(t)$ denotes the Euklidean distance from a to b at time t.

if
(*stationary* a)
(*stationary* b)
(*inside* a b)
($D_{a,b}(t) > D_{a,b}(t+1)$)
then
hypothesize (closer a b).

The following *constraint*-rule verifies that a hypothesis generated in the previous rule can be maintained over time. Constant vehicle motion is implicitly assumed.

Rule *GEOMETRY-4*: Due to the monotonic functions involved in the imaging process, motion between two (stationary) image points a,b must continue, once it has started.

if
(*stationary* a)
(*stationary* b)
($D_{a,b}(t) > D_{a,b}(t+1)$)
then
verify ($D_{a,b}(t+1) > D_{a,b}(t+2)$).

The results from *geometry* are valid for any arbitrary configuration of points in space. However, certain assumptions can be made about the spatial layout of the scene which will be encountered. For instance, the vehicle always travels on some kind of surface, such that the profile of the view is more or less convex (Fig. 3).

If the camera is fixed to the vehicle, such that the wheels and the ground are below, we can define a (heuristic) rule, that features which are *lower* in the image are generally *closer* to the vehicle.

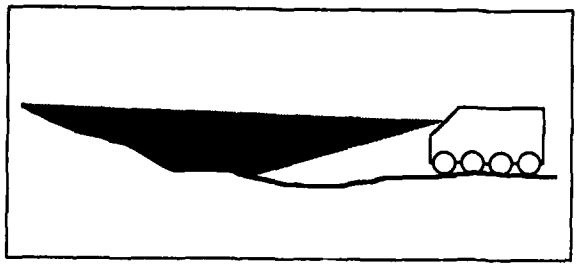


Figure 3. The convex nature of the viewing profile. It allows the heuristic assumption, that features *lower* in the image are generally *closer* to the vehicle.

Rule *SPATIAL-1*: For any pair of image points a,b:
if
(*lower* a b)
then
hypothesize (*closer* a b)

As a consequence, it is very unlikely, that a feature is farther away than all its surrounding neighbors.

Occlusion is another important source of spatial information, which is applicable for more complex features such as lines and regions. A feature occluding another feature is closer to the viewer than the occluded feature.

Rule *OCCLUSION-1*: For any pair of image features (lines, regions) a,b :
if
(*occluding* a b)
then
hypothesize (*closer* a b).

Semantics can supply useful clues, when partial interpretations of the scene are already available. If, for instance, the horizon has been identified, any object above it must be in the sky and cannot be stationary. Similarly, the features of an object recognized as a building would not be considered moving in an ambiguous situation.

All these three sources of information (geometry, spatial reasoning and semantics) contribute to the *creation* or the *elimination* of interpretations in the *Qualitative Scene Model*. The set of rules given above should demonstrate the idea and is far from being complete. Since their underlying assumptions vary considerably, conclusions from different groups of rules carry different weights. Conclusions from *geometry*, which are based on the most general assumptions always outrule results from *spatial reasoning* or *semantics*. Meta-knowledge of this form is used to organize the interaction of the different sets of rules.

THE QUALITATIVE SCENE MODEL

Since the proposed model is vehicle-centered, most information contained in the image is implicitly part of the model. In addition to these *facts*, the other part of the model forms the actual interpretation of the scene. The feature-property *stationary/mobile* and the *closer* relationship between features are the central building blocks of the *Qualitative Scene Model.*

The current set of image features together with assigned properties and mutual relationships constitute an interpretation of the scene. An interpretation is *feasible*, as long as it is free of internal conflicts, e.g. (*closer A B*) and (*closer B A*). The *Qualitative Scene Model* comprises a set of feasible interpretations of the current scene. As the vehicle proceeds, more information about the structure of the scene is accumulated.

Figure 4 shows an example, how a network of *closer*-relationships is constructed in the case of a completely stationary set of features a,b,c,d. Initially nothing is known about their spatial relationships, except that (since they are visible) all are in front of the image plane. This situation is reflected by the graph and the set of facts in Fig. 4a. Since facts of the form (*closer* O a) are implicit, they are only listed for the initial case. As more findings are added to the list of facts, the tree of *closer*-relationships keeps growing, as shown in Fig. 4b-d.

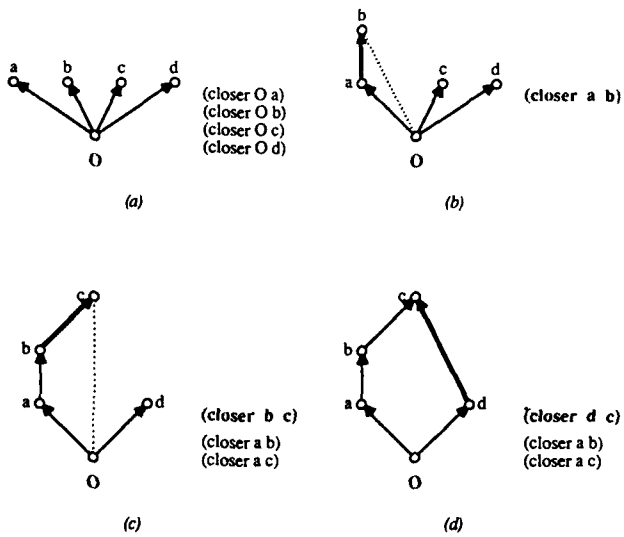


Figure 4. Successive construction of *closer* relationships. *(a)* Initial situation: no relative depth is known for the image features a,b,c,d. They are only known to be in front of the image plane O. *(b)* (*closer* a b) has been determined and is added to the list of facts. *(c)* (*closer* b c) has been determined, (*closer* a c) is implied by transitivity. *(d)* (*closer* d c) has been determined; notice that at this point nothing can be said about the relationship between (a,d) and (b,d).

3. INTERPRETATION AND CONFLICT RESOLUTION

Using a set of rules which include those described in the previous section, the *Qualitative Scene Model* is constructed. In this section we want to describe how the model develops over time, how new interpretations of the scene are generated, and how conflicts are resolved.

Let us consider a simple example with a scene containing only three feature points a,b,c (see Figure 5).

Initially (at $t=t_0$) nothing is known about the spatial relationships between these points and whether they are stationary or not. The default assumption is that any point is stationary, unless there is an indication that this is not true. The initial interpretation of the scene thus contains only

Interpretation A(t_0):
(*stationary* a)
(*stationary* b)
(*stationary* c)

Suppose that between t_0 and t_1 all three points show some amount of expansion away from the FOE, giving rise to the conclusion (e.g. by rule *Geometry-2*) that a is closer (to the vehicle) than b, a is closer than c, and c is closer than b. From the information gathered up to this point, the interpretation of the scene at time t_1 looks like this:

Interpretation A(t_1):
(*stationary* a)
(*stationary* b)
(*stationary* c)
(*closer* a b)
(*closer* a c)
(*closer* c b)

At time t_2 one of the rules claims that c is closer than a and tries to assert this fact into the current interpretation. Clearly, the new interpretation would contain the conflicting facts

(*closer* a c) and (*closer* c a) ,

which would not be a feasible interpretation. The conflict is resolved by creating two disjunct hypotheses, with either a or c as mobile:

Interpretation B(t_2):
(*mobile* a)
(*stationary* b)
(*stationary* c)
(*closer* c b)

Interpretation C(t_2):
(*stationary* a)
(*stationary* b)
(*mobile* c)
(*closer* a b)

Notice, that when a feature is hypothesized to be *mobile*, all its *closer*-relationships are removed from the interpretation. At this point in time, *two* feasible interpretations of the scene are active simultaneously. All active interpretations are pursued until they enter a conflicting state, in which case they are either branched into new interpretations or removed from the *Qualitative Scene Model.*

In our example we assume, that both interpretations B and C are still alive at time t_i. At this point some rule claims, that b should be closer than c. This would not create a conflict in interpretation C, because there c is considered as *mobile* anyway. Interpretation B, however, cannot ignore this new finding, because it contains the contradiction (*closer* c b)! Again we could branch interpretation B into two new interpretation, with either (*mobile* b) or (*mobile* c). This time, however, there is another active interpretation (C), which could absorb (*closer* b c) without causing an internal conflict. Thus interpretation B is not branched out, but removed altogether from the model.

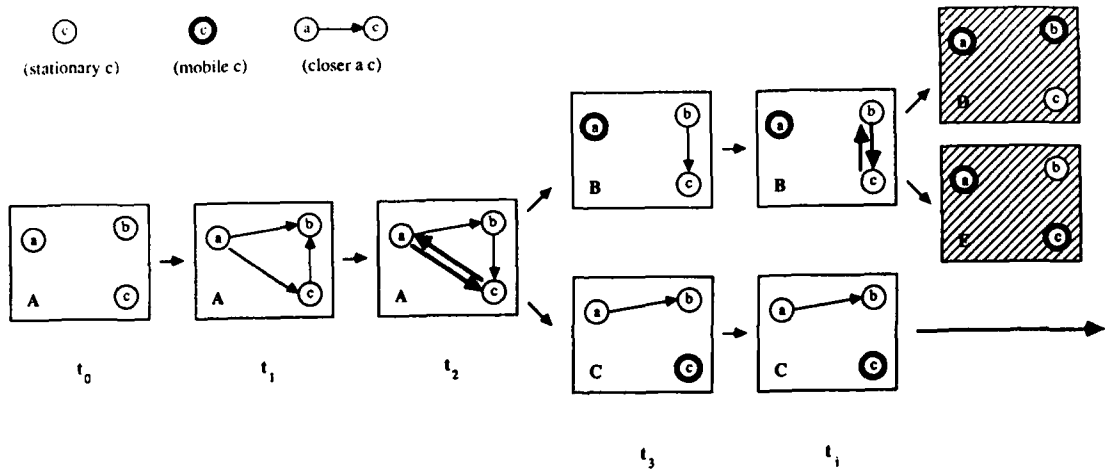


Figure 5. Development of a scene model over time: At time t0 three features a,b,c are given, which are initially assumed to be stationary. At time t1 three *closer*-relationships have been established between a,b,c. All features are still considered stationary. At time t2 a conflict arises in interpretation A, which is the only interpretation active at this time. Two new interpretations (B and C) are created, each containing one feature considered *mobile* (a and c). At time ti another conflict occurs in interpretation B. Since an interpretation (C) is active at the same time, which could absorb the conflicting fact (*closer* c b), B is collapsed and C remains as the only feasible interpretation.

In summary, the development of the *Qualitative Scene Model* is controlled by the following *meta*-rules:

- Initially assume that all features belong to stationary objects. There is only one initial (root) interpretation.
- After a hypothesis has been created by one of the analyzing rules, try to integrate this hypothesis as a fact into every active interpretation.
- If the new fact is consistent with the interpretation, make it a part of this interpretation.
- If the new fact is *not* consistent with the interpretation, and there are currently no other active interpretation, then create a new set of interpretations containing this fact without conflict.
- Otherwise delete this interpretation from the model.

The algorithm for finding the FOE and derotating the image requires a set of image points, which are *believed* to be stationary. Out of all interpretations active at the same time, the *most conservative* interpretation (i.e. with the smallest set of stationary features) is selected for this purpose. Also, the information contained in the *Qualitative Scene Model* is made available to processes such as vehicle control, navigation, object recognition, motion description and threat handling. The main role of the *Qualitative Scene Model* is to serve as a representational bridge between low- and high-level vision and to provide a stable reference over time.

4. CONCLUSIONS

The main difficulty of motion understanding in the ALV scenario is that the effects of vehicle motion and actual movement of objects in the scene contribute to a constantly changing camera image. In this paper we presented a new approach for this problem which departs clearly from previous work. Avoiding a purely numerical technique we advocate a qualitative line of reasoning and modeling. Multiple interpretations of the dynamics in the scene are pursued at the same time, allowing a very flexible way of reasoning. No attempt is made to reconstruct the three-dimensional structure of the scene in numerical quantities. Instead, scene description, motion detection and occlusion analysis are accomplished by reasoning based on a *Qualitative Scene Model*. Different sources of visual information, such as motion, spatial reasoning and semantics are integrated in a rule-based framework. We have tried to show, that this qualitative strategy of reasoning and modeling can provide significant advantages over previous, purely numerical methods.

The work reported here shows the conceptual outline of our approach. While we have considered only point-features so far, the integration of lines and regions will be a natural extension. An implementation on a *Symbolics 3670* is currently under way, using real ALV imagery to demonstrate the benefits of our approach.

REFERENCES

1. B. Bhanu and W. Burger, "Approximation of Displacement Fields Using Wavefront Region Growing," University of Utah, Dept. of Computer Science UUCS 86-111 (June 1986).
2. S. Bharwani, E. Riseman, and A. Hanson, "Refinement Of Environmental Depth Maps Over Multiple Frames," Proc. IEEE Workshop on Motion, Kiawah Island Resort (May 1986).
3. T. J. Broida and R. Chellapa, "Kinematics and Structure of a Rigid Object from a Sequence of Noisy Images," Proc. IEEE Workshop on Motion, Kiawah Island Resort (May 1986).
4. J. Q. Fang and T. S. Huang, "Some Experiments on Estimatiing the 3-D Motion Parameters of a Rigid Body from Two Consecutive Image Frames," *IEEE Trans. Pattern Analysis and Machine Intelligence* **PAMI-6**(5) pp. 545-554 (September 1984).
5. G. Hagert, "What's in a mental model? On conceptual models in reasoning with spatial descriptions," Proc. 9th International Joint Conference on Artificial Intelligence, Morgan Kaufmann Publishers (1985).
6. E. C. Hildreth and N. M. Grzywacz, "The Incremental Recovery of Structure from Motion: Position vs. Velocity Based Formulations," Proc. IEEE Workshop on Motion, Kiawah Island Resort (May 1986).
7. R. Jain, S. L. Bartlett, and N. O'Brien, "Motion Stereo Using Ego-Motion Logarithmic Mapping," Proc. IEEE Conf. Computer Vision and Pattern Recognition (1986).
8. B. Kuipers, "Qualitative Simulation," *Artificial Intelligence* **29** pp. 289-338 (1986).
9. D. N. Lee, "The optic flow field: the foundation of vision," *Phil. Trans. R. Soc. Lond. B* **290** pp. 169-179 (1980).
10. H. C. Longuet-Higgins and K. Prazdny, "The interpretation of a moving retinal image," *Proc. R. Soc. Lond. B* **208** pp. 385-397 (1980).
11. A. Z. Meiri, "On Monocular Perception of 3-D Moving Objects," *IEEE Trans. Pattern Analysis and Machine Intelligence* **PAMI-2**(6) pp. 582-583 (November 1980).
12. A. Mitiche, S. Seida, and J. K. Aggarwal, "Determining Position and Displacement in Space from Images," Proc. IEEE Conf. Computer Vision and Pattern Recognition (June 1985).
13. H.-H. Nagel, "Image Sequences - Ten (octal) Years - From Phenomenology towards a Theoretical Foundation," Proc. Intern. Conf. on Pattern Recognition, Paris (1986).
14. K. Prazdny, "Determining the Instantaneous Direction of Motion from Optical Flow Generated by a Curvilinear Moving Observer," *Computer Graphics and Image Processing* **17** pp. 238-248 (1981).

15. K. Prazdny, "On the Information in Optical Flows," *Computer Vision, Graphics, and Image Processing* **22** pp. 239-259 (1983).

16. D. Regan, K. Beverly, and M. Cynader, "The Visual Perception of Motion in Depth," *Scientific American*, pp. 136-151 (July 1979).

17. J. H. Rieger, "Information in optical flows induced by curved paths of observation," *J. Opt. Soc. Am.* **73**(3) pp. 339-344 (March 1983).

18. H. Shariat and K. E. Price, "How to Use More Than Two Frames to Estimate Motion," Proc. IEEE Workshop on Motion, Kiawah Island Resort (May 1986).

19. R. Y. Tsai and T. S. Huang, "Uniqueness and Estimation of Three-Dimensional Motion Parameters of Rigid Objects with Curved Surfaces," *IEEE Trans. Pattern Analysis and Machine Intelligence* **PAMI-6**(1) pp. 13-27 (January 1984).

20. S. Ullman, *The Interpretation of Visual Motion*, MIT Press, Cambridge, Mass. (1979).

21. S. Ullman, "Maximizing Rigidity: The Incremental Recovery of 3-D Structure from Rigid and Rubbery Motion," MIT A.I.Memo No. 721 (June 1983).

22. Y. Yasumoto and G. Medioni, "Experiments in Estimation of 3-D Motion Parameters from a Sequence of Image Frames," Proc. IEEE Conf. Computer Vision and Pattern Recognition (1985).

23. X. Zhuang and R. M. Haralick, "Rigid Body Motion and the Optic Flow Image," IEEE Conference on AI Applications (December 1984).

What Is A "Degenerate" View?

John R. Kender[1]
David G. Freudenstein

Department of Computer Science
Columbia University
New York, New York 10027

1 Abstract

In this paper, we attempt to quantify what is meant by the terms "degenerate view", and its relatives, "characteristic view", "visual event", and "general viewing position". We propose that the definition of degeneracy is itself degenerate, taking on differing meanings at different times. We claim (at least for the case of polyhedra) that one can only speak of a two-dimensional stimulus as being degenerate *with respect to* a given heuristic for inverting the image function. Additionally, we show that given the finite viewing resolution of a two-dimensional retina, in practice the concept of a characteristic view is often not characteristic of real imagery. Even precisely defined general viewing positions are sensitive to camera acuity: any viewpoint ceases to be characteristic at some resolution, and non-characteristic views are not vanishingly improbable. We provide initial quantitative estimates on these probabilities for some simple cases, and relate them to a minimal disambiguation distance. It follows that an aspect graph is less a discrete graph, and more properly a partitioning of the surface of the viewing sphere into "fuzzy" regions of non-zero area: an aspect *map*. This viewpoint is more in keeping with recent and proposed work on optimal viewing strategies.

2 Introduction

Robotic vision systems must both obtain images and analyze them. However, a primary characteristic of many realistic imaging situations is that the data acquisition is much less costly than the subsequent data analysis. In such domains it is therefore reasonable to dedicate significant computational effort towards the task of calculating an optimal viewing point for the next image capture. Defining and obtaining this optimum is necessarily probabilistic; it must incorporate an understanding of the limits of resolution of the camera, and of the limits of resolution of the placing agent. The overall goal is to obtain maximal information from a sequence of inexact images in inexact placements, while minimizing some work function which expresses the relative costs of image acquisition and image analysis. Such calculations necessarily place a heavy premium on avoiding what are often referred to as "degenerate views".

[1]This research was supported in part by ARPA grant #N00039-84-C-0165, by a NSF Presidential Young Investigator Award, and by Faculty Development Awards from AT&T, Ford Motor Co., and Digital Equipment Corporation.

Nevertheless, it is not apparent what makes a view degenerate, how such a view is recognized or forecast, or even whether such views are rare or commonplace. Thus, the first concern of this paper is to define and quantify the meaning of the term "degenerate", and to show the varying imaging contexts in which it can arise. Secondly, we suggest a representation useful for calculating the likelihood of such views (whatever their definition); it takes the form of mappings over the viewing sphere. This representation extends existing work on aspect graphs by explicitly incorporating the known limits of visual acuity. It also leads directly to methods of associating with each view a probability of its being attained, and the placement cost of attaining such a view.

Therefore, as a first step to a general theory of sensing strategies for finite resolution imagery, we use these aspect *maps* to quantify degeneracy. Although this paper is limited to an analysis of the issue from the perspective of two-dimensional (areal) imagery, it is clear that related problems occur with other types of sensors, whether they are point-, line-, or area-based, and whether they sample distance, orientation, reflectance, motion, or combinations of them.

3 Compact and Partial Survey of Existing Approaches

How to model the viewpoints of a three-dimensional object is related to the general question of three-dimensional modeling [Requicha 80]. Most schemes have traditionally been classified as either "Boundary Representations" (which represent the enclosing surfaces of the object) or "Constructive Solid Geometry" (the boolean combination of volumetric primitives). [Ballard and Brown 82] classify as a third category those representations based on sweeping a two-dimensional region along some specified space curve, although one might think of these "generalized cylinder" representations as an extension of Constructive Solid Geometry.

A priori, there will be an *infinite* number of viewpoints from which a given object will be visible from a fixed distance. The "characteristic view" representation attempts to solve this problem by partitioning the infinite views into a *finite* set of

viewpoint classes. Each viewpoint class represents *all* of the (infinite) various viewpoints from which a specific set of faces of the object will be visible[1].

Thus, an implicit objective of the "characteristic view" representation of a three-dimensional object is not only to encode the object's three-dimensional structure using a type of boundary representation, but explicitly to represent the possible equivalence classes of images obtained by a camera at any of the infinite possible viewpoints. Such information is rich in the sense that it relates directly to the procedural information necessary to instantiate models and to determine future viewpoints.

One of the more successful three-dimensional vision systems to date is the ACRONYM system [Brooks 81], which combines information from all levels of computer vision. Binford describes the system as "generic with respect to observation, that is, ACRONYM is insensitive to viewpoint and flexible with varied sensor inputs." ([Binford 82], p. 38). However, it is worth noting that the success of ACRONYM has been achieved without the need to compare images obtained from different viewpoints of the same object (other than stereo pairs taken from a "characteristic" position: directly above the basically horizontal objects).

Several schemes for representing the set of possible "characteristic views" for a given object have been proposed [Ansaldi, De Floriani, and Falcidieno 85; Callahan and Weiss 85; Wong and Fu 84; Fekete and Davis 84]. Several of these have dealt explicitly only with smooth objects such as tori. One important question which has not yet been resolved is whether the ideal representation should be general enough to cover both smooth objects and polyhedra, or whether it is more efficient to use different schemes for both. Of course, in the latter case, it remains further to be determined how to integrate such representations in imagery containing both types of objects simultaneously.

[Callahan and Weiss 85] use a graph embedded on the viewing sphere as a model for surface shape. They present many useful examples of their model, although they describe only smooth surfaces. Borrowing from [Koenderink and van Doorn 79], visual "events" are clearly encompassed by the representation, and analogues to "typical views" for several objects (bumpy sphere, tori, etc.) are presented. However, there is no apparent connection between their formalism and image observables; many of the examples imply semi-transparent objects.

One recent approach [Fekete and Davis 84] apparently works for either polyhedral or smooth objects. A new data structure ("hierarchic trixels") for representing a function on the viewing sphere is presented, which explicitly encodes a discrete sampling of the values of an arbitrary function of the image, measured on images generated from all possible viewpoints. Object recognition is performed based on this set of function values. This resulting function on the sphere, the "property sphere", can be used to generate aspect graphs.

It is a significant result of this work that the two-dimensional analog of the property sphere, the "property circle", has proved sufficient for useful object recognition tasks. Another noteworthy aspect of the work of [Fekete and Davis 84] is that the arbitrary function used until now has (intentionally) been a metric of relatively low information content, e.g. the average radius of the image from the given viewpoint. This metric reduces area information in an image to a small number of scalars (here, one), in effect turning the camera into a point-based sensor. It remains to be seen whether other intuitively "higher" information-content metrics will perform better. In spirit, the property circle work has much in common with the work of Grimson on sensing strategies for point-based sensors [Grimson 85], where the concept of trixel has as its analogue the Chebyshev point of the bounding polygon of a projected object surface.

The property sphere concept has been extended by [Korn and Dyer 86], who specify a number of useful algorithms for constructing and manipulating the property sphere representation. Specifically, in order to minimize redundant data, Korn and Dyer have extended the concept of *2-D* region growing [Ballard and Brown 82]. This method saves both space and time in further operations, by merging "viewpoints into maximally connected regions such that all viewpoints of a region have the same value of a given property *P*" (p.22, *ibid.*). An analysis is presented of a particular implementation of the property sphere.

The viewpoint-modeling approaches mentioned so far are motivated by the ultimate goal of performing *object recognition*. However, similar viewpoint models are important for another purpose: planning camera positions. Planning future viewpoints will, of course, be part of most object recognition schemes. Nonetheless, it may be viewed as a distinct *subgoal* of *object recognition*, and sometimes as a totally independent task.

Thus, in contrast to the approaches mentioned so far, [Canny 84] does not explicitly model the various viewpoints of a given object. Rather, his goal is to characterize "the regions in space from which a given feature" is either entirely or partly visible. Working in the context of *3-D* mechanical part inspection, Canny deals with the question of planning camera positions.

4 Degenerate Viewpoints in Theory

If it is to be useful, a representation for the views of a three dimensional object or object assembly must give some insight into those viewing positions which are less helpful in resolving ambiguities of object structure, position, or orientation. We present two common views of what such degeneracy is, show that they are deficient, and redefine them in ways that are more quantifiable.

For now, we will approach the problem of image degeneracy purely theoretically, under the assumption of a camera with infinite resolution under orthographic projection. Later we will make resolution finite; and although we will not address the issue directly, we will occasionally allude to the problems of perspective projection.

4.1 Slight Movements Giving Drastic Changes?

Perhaps the simplest example of ambiguity is the case of a head-on view of a cube, which is ideally imaged as a square. Such an image has often been noted as giving no information as to the three-dimensionality of the object, and has therefore

[1] A recent application of this concept to 3D object representation in the context of bin-picking tasks is reported in [Ikeuchi 86]. In particular, an interpretation tree is generated which distinguishes among viewpoints based on the set of visible faces.

been described as a degenerate image (see for example, [Kanade 80; Sabbah 82]). Degeneracy in this context, however, refers to the fact that a "slight" change in the viewpoint which generated the image would cause a "drastic" change in the image (sometimes called an "image event").

This definition (and related descriptions of what makes a viewing position general or an image characteristic [Chakravarty 82]) is inadequate in two ways:

1. It is vague with respect to the meanings of "slight" and "drastic".

2. It does not encompass all the phenomena that would seem to be properly described as examples of degeneracy.

What changes drastically in the cube-as-square image can be characterized in many ways: the number of regions change, the topology of the image regions is altered, apparent symmetries are modified, and (if lines are labeled in the Huffman-Clowes manner) the junctions are relabelled (cf. [Lavin 74; Thorpe and Shafer 83]). Still other derived properties of the image change, too. More generally, depending on the means of analysis, this view of the cube would be called degenerate if a slight change in viewpoint would alter the "quality" of the ensemble of extracted image features used in shape analysis. A rigorous definition of this "quality" change must then include the requirement that a qualitative change is one that ultimately affects the derivation of those object's "semantic" properties (such as identity, scale, rotation, coloring, etc.) considered important by the system.

The drastic change is therefore a drastic change in *interpretation*, not in image. Therefore the perception of a drastic change can vary from system to system. For example, if the system distinguishes cubes from spheres by detecting the presence or absence of long straight lines, the cube-as-square cannot be considered degenerate. (And, in fact, if the cube is the only model in the system at all, no view is ever degenerate.)

What is hiding behind this implicit definition of "drastic" is the interpretation equivalence relation; a drastic change is a change of interpretation equivalence classes. However, since in many systems the interpretation classes are inherited from the feature equivalence classes, commonly the drastic change has been attributed to image characteristics alone.

Similarly, the notion of "slight movement" is imprecise. What is usually implicit in such definition is that there is at least one direction in which an arbitrarily small movement of the camera causes the drastic change. (Usually the direction is on a line perpendicular to an image edge). The meaning of "arbitrarily small" only appears to make sense when taken in the sense of mathematical analysis. That is, the drastic change must occur for positive movements of magnitude less than some epsilon, in the direction of degeneracy.

Such a definition would imply that degeneracy can be qualified as a matter of varying degree, although this is apparently never stated. That is, what can vary from degeneracy to degeneracy is the number of possible ("qualitative") drastic changes, and the relative number of directions in which such resolutions (or non-resolutions) occur. For example, the cube-as-square image can resolve itself into an image with either two or three regions, with the two region image possible only for four discrete directions of camera movement. All other directions resolve it into an image with three regions, even if some of those regions are vanishingly thin. Further, some "degenerate" views sometimes do not resolve at all. For example, the cube imaged as two rectangles remains two rectangles for two discrete directions of movement, resolving itself into three regions under movement in all other directions. The space of allowable degeneracies is apparently very large, and perhaps can be quantified in absolute terms as some measure defined on the ways in which the view fails to resolve into something more "characteristic".

The converse of the common "small gives drastic" definition is perhaps easier to implement. This converse definition is stated as follows: An image is seen from a "general viewpoint" if there is some positive epsilon for which camera movements in *any* direction can be taken without effect on resulting semantic analysis. Here, too, the definition is based ultimately on system performance; an image can be degenerate to one system but not another. Note that this definition of degeneracy need not directly appeal to any consideration of three dimensional models.

4.2 Unlikely Views?

However, even this definition of a degenerate viewpoint is incomplete; basically it says that generality is a form of stability. However, many viewpoints are stable yet ought to be considered degenerate, at least in the sense that that they are less likely to allow a system to instantiate a proper model than other viewpoints do.

Consider a pyramid with a square base and arbitrary height. (Customarily, it has equilateral triangles for its sides, but we relax that restriction.) Imaged from many viewpoints from below, its image appears to be a type of rhomboid: a tilting square. None of these viewpoints is degenerate according to the stability definition above, since a slight change in viewpoint does not cause a drastic change in the image; it merely tilts the rhomboid. In fact, there is a great deal of viewpoint freedom, and many views appear to yield the same semantic result: a partly instantiated pyramid with height information largely missing. What is most disturbing about such views is that they are potentially the most common. For a very flat pyramid, such rhomboids appear from nearly half of all viewing directions.

Yet it seems plausible to suggest that these particular views be considered at least partially degenerate; in contrast to some other views, these images give little information about how to instantiate the pyramid's height. (They do place weak upper limits on the height: the peak is constrained to heights that keep it invisible.) Further, if our model base were more complete, we would not be able to distinguish such a view from among similar views of a triangular wedge with square base, or any similarly tapering polyhedron with a square base, despite the stability of our vantage point. Thus, we may wish to include in our definition of degeneracy those viewpoints from which "relatively little" three-dimensional information may be obtained, regardless of stability.

Operationally, this aspect of degeneracy can be quantified as the expected number of additional views necessary to disambiguate the object; a degenerate view is therefore one that is relatively uninformative and will require more images. This number clearly depends on the complexity of the model data base, the intelligence of the system procedures for

determining the "best" next view, and the (necessarily) heuristic procedures of the system for inverting the image projection. Again, this is a system performance definition, and an image's degree of degeneracy would change as system parameters change.

It would appear that this definition must be probabilistic. It is not hard to conceive of objects or object assemblies in which multiple viewpoints give identical images, but for which the resolution into a single interpretation takes varying strategic paths depending on the differing image features that can appear in the second view. (Take as an example a cube with a single distinguished face, viewed initially so that only one non-distinguished face is visible.) The likelihood of taking each path can be quantified: the most inclusive measure of an image's degeneracy would be then be its probability-labeled search tree. Various measures based on the full tree (one of which is, of course, is expected depth) could also serve as a measure of degeneracy. Under this definition, tree breadth has no strong role: a degenerate image can resolve itself into hundreds of images, but as long as each new image was interpretable, the original image is no less degenerate than the pyramid viewed from below.

It is interesting to note a paradoxical consequence of this systems' view of degeneracy. As a system's power increases due to the availability of more sophisticated shape analyzing tools (such as when shape from skewed symmetry is used with shape from shading), more types of ambiguity are possible. Each method brings with it a weakness. The implication is that vision systems with multiple sources of knowledge must know when to ignore a source undergoing degeneracy. This meta-knowledge can be explicitly coded, or *implicitly* handled by means of a flexible enough representation that permits "don't know *much*" as a valid answer.

5 Specific Imaging Degeneracies

Considering now only images of polyhedral objects, it is possible to give a catalogue of image degeneracies. Each is based on a specific heuristic for inverting the three-dimension to two-dimension image function. The list is partial, and omits some heuristics that are even more fundamental, such as the generally assumed heuristic rules that lines in the image have been caused by lines in three-space. (In this last case, this would imply that any planar curve imaged from within its plane would often be considered as degenerate, if the system were unable to interpret it as other than a linear object.)

5.1 Vertices Imaged in the Plane of Scene Edges

Apparently the major source of "degenerate views" in the blocks world (see Figure 1), so-called coincidental alignments occur when the image of a vertex appears to fall on the image of an edge. They confound the basic imaging assumption that three or more lines coincident in the image are coincident in the scene. If the scene is analyzed using labelling, the labelling will fail. The image is then degenerate because another image is required. In theory, such coincidental alignments have probability zero, since the camera must lie on a specific plane (or more precisely, in the infinite intersection of two co-planar half-planes).

[2]see Figure 5.

5.2 Parallel Scene Lines Imaged in Their Own Plane

This is one of the degeneracies observed with the cube[2]. It violates the heuristic that colinear in the image implies colinear in the scene, a heuristic often not used. Hence, it is system-sensitive. In theory, it also has probability zero, although the camera placement is somewhat more free than in the case of vertex-on-edge.

5.3 Coincident Scene Lines Imaged in Their Own Plane

This is a special case violation of linear in the image implies linear in space. Again, this is system dependent and has probability of zero.

Figure 6 illustrates one example of this degeneracy while viewing a pyramid.

5.4 Perfect Symmetry

This is an interesting extreme case, and one apparently avoided by professional photographers as it appears to flatten relief[3]. It is apparently based on the heuristic that symmetry in the image implies symmetry in the scene perpendicular to the line of sight. Analogous to what happens when a cube is imaged as two congruent rectangles, it is often degenerate since perfectly symmetric images lack the cues to depth that broken (skewed) symmetries provide. It has a probability of zero of occurring ideally, although the camera now has the freedom to move in at least one entire plane.

6 The Effect of Finite Resolution on Specific Imaging Degeneracies

The various viewing points for our camera may be modeled as points on a viewing sphere at whose center lies the object of interest. *Therefore, in the* ideal case of the cube with infinite resolution under orthography (or, for that matter, under perspective) there are precisely six viewing directions from which we see exactly one face of the cube and no more. Similarly, a family of three mutually orthogonal great circles which intersect at these six points determine the set of directions from which we would see exactly two faces of the cube. Anywhere else on the sphere we see three faces (see Figure 2). A point on the sphere chosen at random will be a viewpoint imaging three faces with probability 1.

Of course, any real system will have only finite resolution. How this resolution is measured, and how repeatable it is, can vary depending on application. For the cube, resolution appears to be the ability to separate two nearly concurrent parallel lines into their separate sources. (Under perspective, the parallel lines would only be nearly parallel.) Assuming this resolvability of parallels is independent of the line segment lengths (admittedly, this is somewhat unrealistic), then the zero probabilities of degeneracy become finite. On the viewing sphere, the great circles have become bands, and the points of single face viewing have become spherical squares. (See Figure 3). Their relative areas (and hence the probability of degeneracy) are straightforward to compute in terms of camera acuity. The less accurate the camera, or the farther it is away, or the smaller the object, the larger the likelihood that a viewpoint is degenerate. In the extreme, the bands merge, and no viewpoint sees a "characteristic" view: the images are infinitely degenerate.

[3]we provide a straightforward example of perfect symmetry in Figure 7.

The edges of the bands, however, cannot be sharp. Although the bands partition the surface of the viewing sphere, their borders represent those viewing directions at which parallel lines are "first" seen as two lines. Given camera inaccuracy and noise, this transition to resolvability cannot be sudden. Depending on the camera and the accuracy of the algorithms processing its data, repeatability may best be represented by a fuzzy boundary. Thus instead of each point on the sphere having a label, it has a vector of (label, likelihood) pairs. Each degeneracy region, then, fades away in likelihood as it extends farther from its ideal point or great circle. The actual computation of the shape of this probability density depends on the image heuristic used for image function inversion, as well as some measure of camera and software precision. Nevertheless, the relative size of the integral of probability over the partition can give an accurate estimate of the likelihood that a particular view, degenerate or not, will be visible after a random camera placement.

Thus:

6.1 Vertices Imaged in the Plane of Scene Edges

The system must separate vertices from lines. Assuming a basic radius of confusion around the vertex, under orthography the degeneracy region on the sphere is a band segment. The band width is inversely proportional to accuracy, but the band position on the viewing sphere and its angular extent are determined by the geometry of the scene. Basically, the band segment lies in the infinite intersection of two co-planar half-planes determined by the vertex in the scene and the line in the scene. Thus, the viewing sphere of a polyhedral assembly of object is criss-crossed with such bands; there are approximately as many as there are vertices times the number of edges. (Not quite true; both the vertex and the edge must be visible.) Degenerate views are therefore rather common in practice, and become more so as the camera recedes.

6.2 Parallel Scene Lines Imaged in Their Own Plane

Again, this is the case of the finite resolution cube, assuming that parallelism detection is independent of line length. The degeneracy region is a band. This is likely to be a common source of degeneracy if the imaging environment responds strongly to gravitational influences: there will be many parallel vertical lines.

6.3 Coincident Scene Lines Imaged in Their Own Plane

This source of degeneracy depends on the accuracy of an angle detector. The region of degeneracy is again band-like, with width depending on the detector, and position depending on the scene; the center of the band lies in the same plane as the lines. The exact configuration is heavily dependent on actual system performance. It may be a common source of degeneracy in the blocks world.

6.4 Perfect Symmetry

Again, this is heavily dependent on detector performance, but it probably appears as a fairly wide (but less probable) band centered over the axis of symmetry. The relatively large width comes from the fact that symmetry is a global property and its axis is therefore hard to localize.

6.5 Comments

In a sense these aspect maps are property spheres, where the property is a type of degeneracy. Computing them demands substantial computational time and storage [Besl and Jain 85]; both would benefit from a hierarchic, trixel-like approach. Note that since the sphere is topologically equivalent to the extended plane, all such maps can be drawn as planar graphs [Werman, Baugher, and Gualtieri 86] . (See Figure 4, where the "standard"aspect graph has been augmented to show all possible transitions out of degenerate views, as alluded to in [Castore 84][4]).

The finite resolution aspect maps of several polyhedral solids appear to be isomorphic to semiregular polyhedron of the class of "Stott" figures, described in [Stott 10]. They have the pleasant property that each has a Hamiltonian circuit [Miller 71], placing a firm lower bound on the complexity of disambiguating an object given its image and some finite resolution image feature.

7 Relative Probabilities of Viewpoint Classes for the Cube

We might, in a given situation, wish to know the probability of reaching a particular class of viewpoint, given a *random* decision as to "where to go next." For the case of the cube, we can obtain the probabilities of 1, 2, and 3-faced views, by using spherical geometry to calculate the relative surface areas on the viewing sphere of the 3 regions described above (square, rectangular, and triangular patches).

An analysis of the cube shows that these probabilities are generally a function of system resolution only, independent of the distance from the object. Systems capable of higher resolution will generally be less likely to yield "uncharacteristic" views, as one would expect. The details of our analysis are presented in Appendix I.

8 Minimal Disambiguation Distance

Transitions between the various regions of the sphere represent what [Koenderink and van Doorn 79] term a "visual event". Only such a transition is capable of yielding qualitatively new information. Thus these transitions clearly represent a useful change of viewpoint, which would be worth paying for in terms of traveling distance.

For the purposes of minimizing the distance traveled in obtaining further images, it will be useful to quantify the minimal distance we must travel on our viewing sphere, to ensure that we will experience a visual event. If we reach a characteristic view from which an image has not yet been obtained, then we will maximize the probability that any

[4] In particular, we have added, without loss of planarity, the direct transitions between regions where 1 face is visible, and regions where 3 faces are visible. We noticed that in their excellent survey on 3D object recognition, [Besl and Jain 85, p. 89] did not show such transitions in their aspect graph of a cube, nor did their analogues appear in the aspect graph of a tetrahedron, presented by Koenderink and van Doorn. For an interesting discussion of the relevance of such transitions to robotic vision, see the remarks of N. Badler in [Castore 84]

degeneracy will be disambiguated by the new information.. In the case of our viewing sphere of a cube, this would mean ensuring that a 3-faced image is obtained.

Using the assumptions of orthographic projection, we can easily quantify this minimum distance. For a fixed viewing-sphere radius of resolution n (i.e. n is an absolute number equal to the number of pixels our object will occupy in the image), the distance we must travel along the viewing sphere is equal to the product of the length of the radius and the arcsin of $(1/n)$ radians[5].

9 Closing Observations and Possible Future Research

Calculating the number of necessary views and the effort to obtain them is a formidable task. In some senses it resembles the design of part feeders [Natarajan 86]: that is, given an unknown position on the viewing sphere, determine what series of camera movements would inevitably lead to a distinguished configuration, namely, the acquisition of all relevant semantic information about an object or object assembly. Even assuming one knows perfectly where one is on the viewing sphere, the determination of even the distance to the nearest visual event is complex, given its probabilistic nature. Circumstances are easy to construct (for example, when the object is too small) where it is actually impossible.

The aspect map can be augmented with other information. It can incorporate probabilities such as the likelihood of a given gravity-induced preferred orientation, or it can be convolved with a placement uncertainty spread function. The spread function can be variable, itself incorporating such information as the robotic work space or other constraints on placement motion. A search for the optimal next view could then also minimize camera placement error and also, by related methods, camera placement costs.

Such algorithms would be particularly valuable if ways exist to formally combine the aspect maps of individual objects to create the aspect map of an object assembly. Thus, from a few primitives and a little knowledge of the robotic placer and its workspace, a single representation could direct active sensing. Whether or not such a representation is ultimately practical, it has nevertheless been helpful in elucidating the meanings of "general viewing position" and "degenerate view".

References

[Ansaldi, De Floriani, and Falcidieno 85]
Ansaldi, S., De Floriani, L., and Falcidieno, B.
Geometric Modeling of Solid Objects using a Face Adjacency Graph Representation.
In *SIGGRAPH '85 Conference Proceedings*, pages 131-139. San Francisco, CA, July 22-26, 1985.

[Ballard and Brown 82]
Ballard, D. H., and Brown, C. M.
Computer Vision.
Prentice-Hall, Englewood Cliffs, NJ, 1982.

[Besl and Jain 85] Besl, P. J., and Jain, R. C.
Three-Dimensional Object Recognition.
ACM Computing Surveys, 17(1):75-145, March, 1985.

[Binford 82] Binford, T. O.
Survey of Model-Based Image Analysis Systems.
International Journal of Robotics Research, 1(1):18-64, 1982.
(Spring issue).

[Brooks 81] Brooks, R. A.
Symbolic Reasoning among 3-Dimensional Models and 2-Dimensional Images.
Artificial Intelligence, 17:285-348, August, 1981.

[Callahan and Weiss 85]
Callahan, J., and Weiss R.
A Model for Describing Surface Shape.
In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 240-245. San Francisco, CA, June, 1985.

[Canny 84] Canny, J. F.
Algorithms for Model-Driven Mechanical Part Inspection.
Research Report RC10505, IBM, November, 1984.

[Castore 84] Castore, G. M.
Solid Modeling, Aspect Graphs, and Robot Vision.
In Pickett, M. S., and Boyse, J. W. (editors), *Solid Modeling by Computers: From Theory to Applications*, pages 277-292. Plenum Press, New York - London, 1984.

[Chakravarty 82] Chakravarty, I.
The Use of Characteristic Views as a Basis for Recognition of Three-Dimensional Objects.
PhD thesis, Image Processing Laboratory, Rensselaer Polytechnic Institute, 1982.
(report number IPL-TR-034).

[Chakravarty and Freeman 82]
Chakravarty, I. and Freeman, H.
Characteristic Views as a Basis for 3D Object Recognition.
In *Proceedings of the Society for Photo-Optical Instrumentation Engineers Conference on Robot Vision, Vol. 336 (Arlington, VA, May 6-7)*, pages 37-45. SPIE, Bellingham, WA, 1982.

[Crawford 85] Crawford, C. G.
Aspect Graphs and Robot Vision.
In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 382-384. San Francisco, CA, June, 1985.
(presented as a poster paper).

[5] In Appendix II, we present the derivation of the minimum *angle* guaranteed to produce a visual event. The distance we must travel along the viewing sphere is obtained by multiplying this angle by the radius of the viewing sphere.

[Fekete and Davis 84]
Fekete, G. and Davis, L.
Property Spheres: A New Representation for 3-D Object Recognition.
In *Proceedings of the IEEE Workshop on Computer Vision: Representation and Control*, pages 192-201. Annapolis, MD, April 30 - May 2, 1984.

[Grimson 85]
Grimson, W. E. L.
Sensing Strategies for Disambiguating Among Multiple Objects in Known Poses.
Memo 855, MIT Artificial Intelligence Lab, August, 1985.

[Ikeuchi 86]
Ikeuchi, K.
Generating an Interpretation Tree From a CAD Model to Represent Object Configurations for Bin-Picking Tasks.
Technical Report CMU-CS-86-144, Department of Computer Science, Carnegie-Mellon University, Aug. 7, 1986.

[Kanade 80]
Kanade, T.
A Theory of Origami World.
Artificial Intelligence, 13(3):279-311, May, 1980.

[Koenderink and van Doorn 76]
Koenderink, J. J., and van Doorn, A. J.
The Singularities of the Visual Mapping.
Biological Cybernetics, 24(1):51-59, 1976.

[Koenderink and van Doorn 79]
Koenderink, J. J., and van Doorn, A. J.
Internal Representation of Shape with Respect to Vision.
Biological Cybernetics, 32(4):211-216, 1979.

[Korn and Dyer 86]
Korn, M. R. and Dyer, C. R.
3-D Multiview Object Representations for Model-Based Object Recognition.
Research Report RC11760, IBM, March, 1986.

[Lavin 74]
Lavin, M. A.
An Application of Line-Labeling and other Scene-Analysis Techniques to the Problem of Hidden-Line Removal.
Working Paper 66, MIT Artificial Intelligence Lab, March, 1974.

[Miller 71]
Miller, J. E.
Transmission of Analog Signals over a Gaussian Channel by Permutation Modulation Coding.
PhD thesis, Columbia University, Department of Mathematical Statistics, 1971.

[Natarajan 86]
Natarajan, B.
Motion Planning and its Dual.
PhD thesis, Cornell University, Department of Computer Science, 1986.
forthcoming.

[Requicha 80]
Requicha, A. A. G.
Representations for Rigid Solids: Theory, Methods, and Systems.
ACM Computing Surveys, 12(4):437-464, December, 1980.

[Sabbah 82]
Sabbah, D.
A Connectionist Approach to Visual Recognition.
PhD thesis, Rochester University, Department of Computer Science, April, 1982.
(available as TR107).

[Scott 84]
Scott, R.
Graphics and Prediction from Models.
In *Proceedings of the ARPA Image Understanding Workshop*, pages 98-106. Science Applications Inc., McLean, VA, New Orleans, LA, Oct. 3-4, 1984.

[Stott 10]
Stott, A. B.
Geometrical Deduction of Semiregular from Regular Polytopes and Space Fillings.
Ver. der Koninklijke Akad. van Wetenschappen te Amsterdam (eerstie sectie), 11(1)1910.

[Thorpe and Shafer 83]
Thorpe, C., and Shafer, S.
Topological Correspondence in Line Drawings of Multiple Views of Objects.
Technical Report CMU-CS-83-113, Department of Computer Science, Carnegie-Mellon University, March, 1983.

[Werman, Baugher, and Gualtieri 86]
Werman, M., Baugher, S., and Gualtieri, J. A.
The Convex Visual Potential.
Internal Memo, University of Maryland, Center for Automation Research, February, 1986.

[Wong and Fu 84] Wong, E. K., and Fu, K. S.
A Graph-Theoretic Approach to Model Matching in Computer Vision.
In *Proceedings of the IEEE Workshop on Computer Vision: Representation and Control*, pages 106-111. Annapolis, MD, April 30 - May 2, 1984.

Appendix I

Probabilities for the 3 Viewpoint Types of the Cube

These probabilities will be calculated by finding the *relative surface areas* on the viewing sphere for the three kinds of regions (see Figure 1). Thus, our first goal is to find the surface areas of the three regions on the viewing sphere.

We start by examining **one** of the 3 orthogonally intersecting equatorial *bands*. To find the surface area of one such band, we use the surface area integral for revolving the graph of a non-negative parametric function around the X axis, where x' and y' are continuous, and x' remains positive:

for $x(t), y(t), t \in [c,d]$

$$S = \int_c^d 2\pi\, y(t)\sqrt{[x'(t)]^2 + [y'(t)]^2}\,dt\ .$$

In our case, we generate a sphere of radius r by revolving about the x-axis the following parametric arc: $x(t) = r\cos(t)$, $y(t) = r\sin(t)$, for $t \in [\frac{\pi}{2}, \frac{\pi}{2} + \theta]$.

Thus, we have the desired surface area, S, as:

$$S = 2\pi\int_{\frac{\pi}{2}}^{\frac{\pi}{2}+\theta} r\,\sin t\,\sqrt{r^2\,(\sin^2 t + \cos^2 t)}\,dt$$

$$= 2\pi r^2 \int_{\frac{\pi}{2}}^{\frac{\pi}{2}+\theta} \sin t\,dt$$

$$= [2\pi r^2 - \cos t\,]_{\pi/2}^{\pi/2+\theta}$$

$\rightarrow$

$$S_{Band} = 4\pi r^2 \sin\theta.$$

For the *square* patches, multiply the above quantity by the ratio $\frac{2\theta r}{2\pi r} = \frac{\theta}{\pi}$, i.e. the ratio of the curved square length to the circumference ("great circle") of the sphere.

Thus,

$$S_{Square} = 4\,r^2\,\theta\,\sin\,\theta\ .$$

For the **probability** that a random viewpoint will be from one of these regions, multiply by 6 the number of such regions, and divide by the total surface area of the sphere, $4\pi r^2$:

$$\boxed{P_{SquarePatch} = \frac{6\,\theta\sin\theta}{\pi}}$$

Now, to get the probability that we are in one of the bands, but *not* in a square patch (i.e. that we see exactly 2 faces), multiply S_{Band} by 3 (the number of such orthogonal bands), subtract the 6 square single-face-view regions[1], and divide by the total surface area of the sphere:

$$\frac{3\,(4\pi r^2\sin\theta) - 6\,(4r^2\theta\sin\theta)}{4\pi r^2}$$

$\rightarrow$

$$\boxed{P_{RectangularPatch} = \frac{3\sin\theta\,(\pi - 2\theta)}{\pi}}$$

Once we have calculated $P_{SquarePatch}$ and $P_{RectangularPatch}$, we can calulate the probability, "$P_{TriangularPatch}$", of being in one of the 8 triangular regions from which 3 faces of the cube will be visible. We simply subtract from 1 the probability of being in any *other* region. That is to say

$$P_{TriangularPatch} = 1 - [\,P_{SquarePatch} + P_{RectangularPatch}\,]$$

$$= 1 - \frac{3\,S_{Band} - 6\,S_{Square}}{4\pi r^2}$$

$\rightarrow$

$$\boxed{P_{TriangularPatch} = 1 - 3\sin\theta + \frac{6\,\theta\sin\theta}{\pi}}$$

[1]since the band and square regions are not disjoint, we must make sure not to count the square patches twice!

Appendix II

Derivation of θ, the Minimum Disambiguation Angle[1]

Looking at line RG (passing through point F):

$$X_C = r\cos\theta\,,$$

$$Y_C = r\sin\theta\,.$$

The slope of line RG is $-(d + r\sin\alpha)/(r\cos\alpha)$, where α is defined such that $\alpha = \frac{\pi}{4} - \theta$, and the y-intercept is d. Hence, the equation of line RG is:

$$y = \left(\frac{-d + r\sin\alpha}{r\cos\alpha}\right) X + d$$

To find X_F, we intersect RG with the line $y = s$:

$$y = s = \left(\frac{-d + r\sin\alpha}{r\cos\alpha}\right) X_F + d\,.$$

This yields

$$X_F = \frac{(d-s)r\cos\alpha}{d + r\sin\alpha}\,.$$

However, for the purposes of this derivation, we shall assume that *object size is much smaller than the viewing distance*, i.e.

$$d \gg s$$

and in turn[2],

$$d \gg r\,.$$

So long as this assumption is correct, we may simplify our equation for X_F as follows:

$$X_F \approx \frac{(d-s)r\cos\alpha}{d}\,.$$

Similarly, looking at line GQ (which continues through point F),

$$X_D = -Y_C = r\sin\alpha\,,$$

$$Y_D = X_C = r\cos\alpha\,.$$

We proceed to find the slope of line GQ and its y-intercept, yielding the equation of line GQE as

$$y = \left(\frac{-d - r\cos\alpha}{r\sin\alpha}\right) X + d\,.$$

Intersecting line GQE with the line $y = s$, we find that

$$X_E = \frac{(d-s)r\sin\alpha}{d - r\cos\alpha}\,.$$

[1] the smallest angle of movement along the Viewing Sphere which will guarantee a visual event while viewing the cube. Note that we assume perspective projection.

[2] since r is defined as $\sqrt{2}s$

Typeset by AMS-TEX

Again, under the assumption of object size being much smaller than viewing distance, this simplifies to

$$X_E \approx \frac{(d-s)r\sin\alpha}{d}.$$

Now we wish to find θ, such that the distance taken up in the image between the 2 cube corners, Q and R, is one pixel. This is, in fact, the fundamental constraint, based upon which we are attempting to determine the disambiguation angle for movement along the viewing sphere which will guarantee the "visual event" of (heretofore hidden) face BC becoming visible in the new position of face QR.

The precise definition of this constraint is that segment YZ in the image plane should take up a fraction of RH inversely proportional to the total number of pixels in the image of the *unrotated* cube side. Thus,

$$X_F - X_E = \frac{1}{n}(2s).$$

Combining this with previous equations yields

$$\cos\alpha - \sin\alpha = \frac{\sqrt{2}}{n}.$$

However, using the definition of $\alpha = \frac{\pi}{4} - \theta$, we can simplify by trigonometric substitutions, in terms of θ, eventually arriving at

$$\cos\alpha - \sin\alpha = \sqrt{2}\sin\theta.$$

Combining with previous results yields the final solution of θ in terms of the system resolution, n:

$$\sin\theta = \frac{1}{n}.$$

□

ACKNOWLEDGEMENTS

The authors wish to thank Jonathan Gross and Eugene Pinsky for their helpful comments and suggestions.

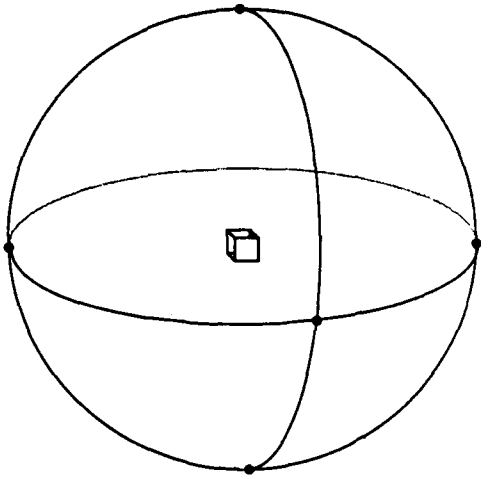


Figure 2
Cube, at center of the "viewing sphere"

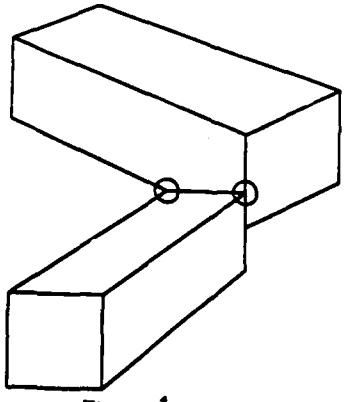


Figure 1
Classical "degenerate" view

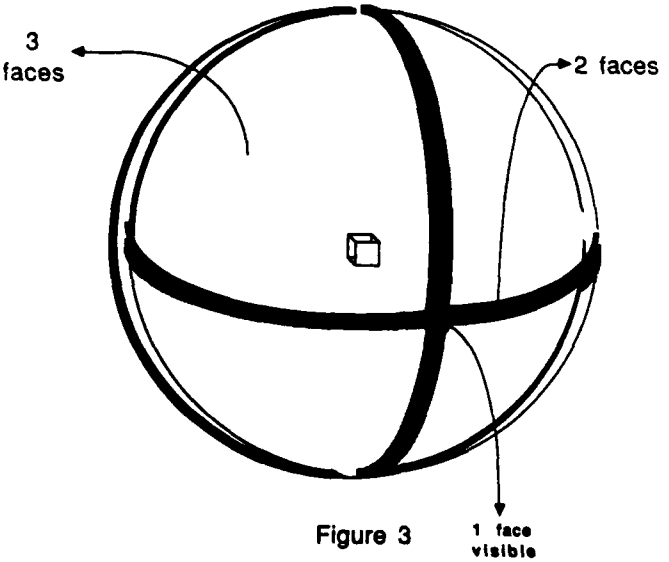


Figure 3
Cube at center of viewing sphere; degeneracies delineated

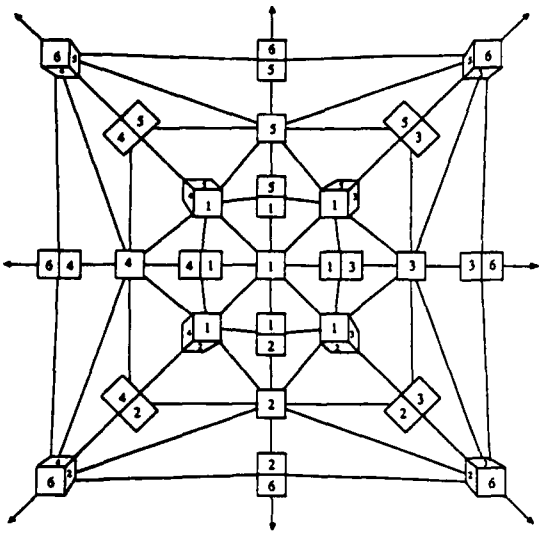


Figure 4

[planar] Aspect Graph for Cube, including 1-face to 3-face transitions

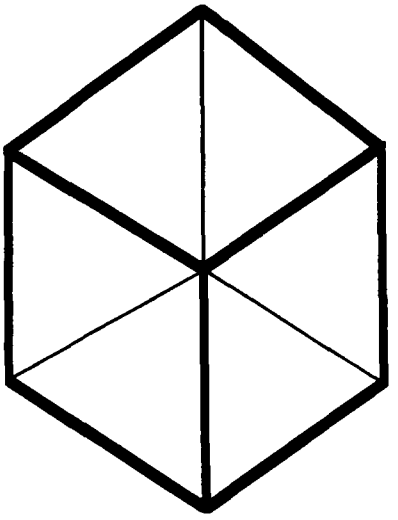


Figure 7

Perfect Symmetry (view of a cube)

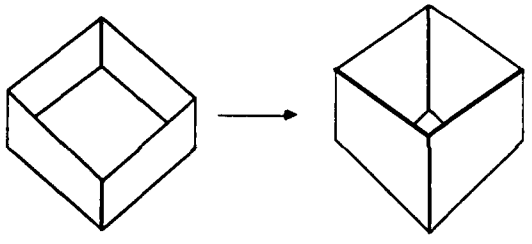


Figure 5

Parallel Scene Lines Imaged In Their Own Plane

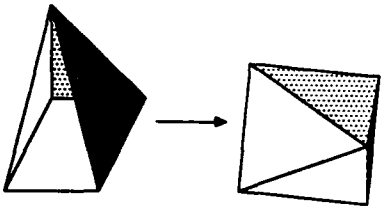


Figure 6

Coincident Scene Lines Imaged In Their Own Plane

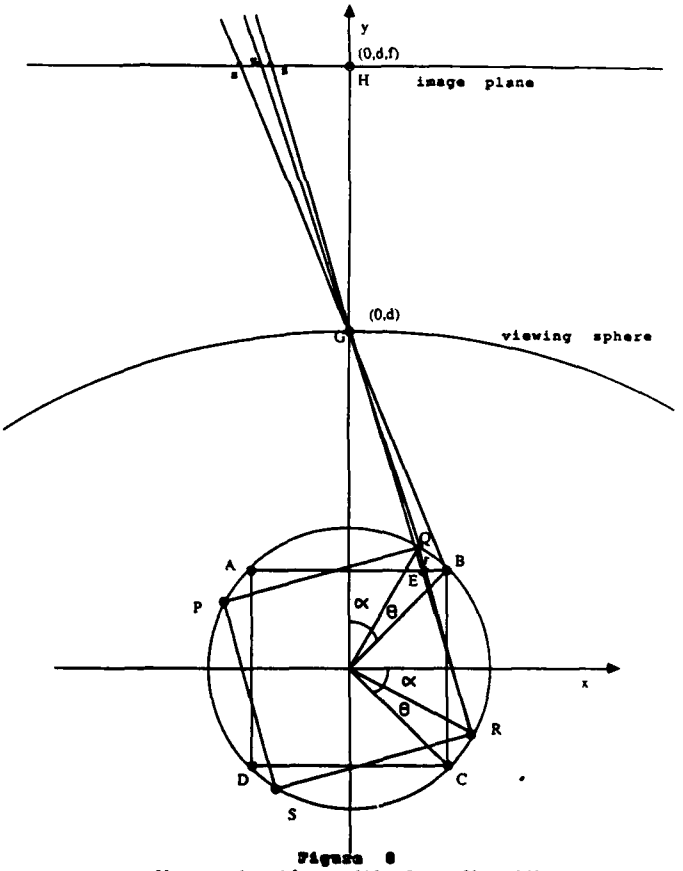


Figure 8
(in conjunction with Appendix II)

THE ROLE AND USE OF COLOR IN A GENERAL VISION SYSTEM

Glenn Healey and Thomas O. Binford

Artificial Intelligence Laboratory
Computer Science Department
Stanford University
Stanford, CA 94305

Abstract

We show that an intelligent approach to color can be used to significantly improve the capabilities of a vision system. This paper is divided into three parts. In part I, we adopt general physical models for the properties of objects which determine how they interact with light. These models are far more general than those typically used in computer vision. From these models we derive generic properties relating color, geometry, and image irradiance. In part II, we discuss the recovery and representation of color. We show how to recover physical color from images captured using different sensors. Under general assumptions, we derive expressions for sensor spectral transmission functions which are provably optimal for color recovery. A metric space for colors is introduced and its significant properties are described. Within this general framework for color, we discuss the advantages of the specific representation employed by our system. In part III, we use our physical models to derive powerful algorithms for extracting invariant properties of objects from images. The first algorithm is used for the generic classification of objects according to material. The second algorithm provides a solution to the color constancy problem. These algorithms have been implemented and consistently produce correct results on real images. Some examples of experimental results are presented.

1. Introduction

A general vision system must be capable of generating meaningful descriptions of a scene in any of the diverse environments for which human vision is useful. Nearly all existing artificial vision systems are special purpose. They rely heavily on domain-specific constraints and special-case engineering. Although significant progress has been made in machine vision, the best artificial vision systems still fall far short of achieving the capabilities associated with general vision.

Any general vision system must possess robust generic models of both the physical world and the image-forming process. For a vision system performing a fixed task in a simple environment, simple domain-specific models are often adequate. Unfortunately, such simple models typically become useless when the system is presented with a slightly more complex environment.

Given robust physical models, a general vision system must be able to use these models to extract invariant properties of objects from images. Invariant properties are those properties which do not depend on either the imaging geometry or illumination conditions. The ability to infer invariant properties of objects allows a vision system to generate meaningful descriptions of a scene in any environment. The human vision system, in particular, is remarkably successful at inferring invariant properties of objects from images in widely varying environments.

In this paper, we analyze the role and use of color in a general vision system. As the first step in the analysis, we examine the physics of reflection to develop general models which describe the formation of color images. From these models, we isolate the invariant properties of objects with respect to color. Once these invariant properties are understood, we derive general procedures which reliably extract these invariant properties from images.

This paper is organized into three parts. Part I (sections 2-4) describes the physics underlying the formation of color images. Part II (sections 5-8) gives methods for color recovery and representation. Part III (sections 9-12) presents our color algorithms and experimental results.

I. PHYSICAL CAUSES OF COLOR

2. The Physics of Reflection

There are several ways an object can modify in-

cident light. An object can change light spatially by reflecting it into a small or large angle. Incident light can be modified spectrally. Also, the energy of incident light can be reduced if a large fraction of the light is absorbed by an object.

In this section, we describe general models for the properties of objects which determine how they interact with light. In part III, we use these models to develop techniques for inferring invariant properties of objects from images.

2.1. Fresnel Reflection

When light is incident on the surface of an object, some fraction of it is reflected. A smooth surface reflects light only in the direction such that the angle of incidence equals the angle of reflection. The properties of this specularly reflected light are determined by the optical and geometric properties of the surface. For materials which are optically homogeneous, appearance is completely determined by the properties of specularly reflected light. Metals make up the largest class of homogeneous materials. For many inhomogeneous materials, such as plastics, specular effects are also significant.

The optical properties of a surface material are summarized by the complex index of refraction $M = n - iK_0$ where n is the refractive component and K_0 is the absorptive component. Both n and K_0 are functions of wavelength. From M, the Fresnel equations completely describe the light reflected from a surface. If unpolarized light is incident at an angle θ_l, then the monochromatic specular reflectance F is

$$F = 0.5(R_\perp + R_\parallel) \tag{2.1}$$

where $R_\perp$ is the perpendicular polarized component and $R_\parallel$ is the parallel polarized component. From electromagnetic theory, $R_\perp$ and $R_\parallel$ are given by

$$R_\perp = \frac{a^2 + b^2 - 2acos\theta_l + cos^2\theta_l}{a^2 + b^2 + 2acos\theta_l + cos^2\theta_l} \tag{2.2}$$

$$R_\parallel = R_\perp \frac{a^2 + b^2 - 2asin\theta_l tan\theta_l + sin^2\theta_l tan^2\theta_l}{a^2 + b^2 + 2asin\theta_l tan\theta_l + sin^2\theta_l tan^2\theta_l} \tag{2.3}$$

where

$$a = 0.5\sqrt{\sqrt{(n^2 - K_0^2 - sin^2\theta_l)^2 + 4n^2K_0^2} + (n^2 - K_0^2 - sin^2\theta_l)} \tag{2.4}$$

$$b = 0.5\sqrt{\sqrt{(n^2 - K_0^2 - sin^2\theta_l)^2 + 4n^2K_0^2} - (n^2 - K_0^2 - sin^2\theta_l)} \tag{2.5}$$

Equations (2.2) and (2.3) are known as the Fresnel equations. The Fresnel equations are derived in many places including [4].

The Fresnel equations describe reflection from a smooth surface. In practice, most surfaces are not smooth. The Torrance-Sparrow specular model [23] describes specular reflection from rough surfaces. This model assumes that a surface is composed of small, randomly oriented, mirror-like facets. Only facets with a normal oriented in the perfect specular direction contribute to the monochromatic specular reflectance R_S. The model also quantifies the shadowing and masking of facets by adjacent facets using a geometrical attenuation factor. The resulting specular model is

$$R_S = FDA \tag{2.6}$$

where

F = Fresnel specular reflectance
D = facet orientation distribution function
A = adjusted geometrical attenuation factor

2.2. Colorant Layer Scattering

For inhomogeneous materials, the most prominent optical process is the scattering of light by colorant layers. Inhomogeneous materials include plastics, paper, textiles, and paints. In this section, we describe a physical model for the scattering of light by inhomogeneous materials.

The fraction of the incident light which is not specularly reflected enters the body of the material. For inhomogeneous materials, the body is composed of a vehicle and many embedded colorant particles. While in the body of a material, light interacts with many colorant particles. When light encounters a colorant particle, some portion of it is reflected. The net result of many reflections is that the light is diffused and a significant fraction exits through the surface in a wide range of directions (Figure 1).

By scattering we refer to this process of diffusion by many reflections. In addition to reflecting light, the colorant particles also selectively absorb certain wavelengths. This selective absorption is responsible for the color of the scattered light. In general, this light will be of a different color than the light reflected from the surface.

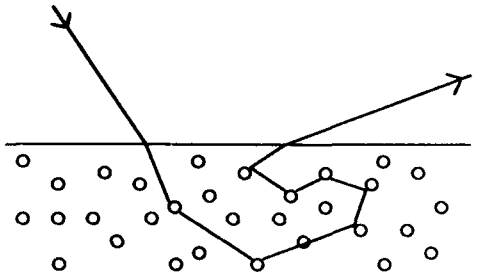


Figure 1. Scattering by Colorant Particles

Kubelka-Munk (K-M) theory [15] is a general mathematical treatment of scattering and absorption in colorant layers. The KM theory assumes that a colorant layer is composed of a large number of optically identical elementary layers. The thickness of each elementary layer is small compared to the thickness of the entire colorant layer, but is large compared to the diameter of individual colorant particles. Thus it is not necessary to model the optical properties of individual colorant particles. The effects of many colorant particles are modeled by the properties of an elementary layer. An elementary colorant layer is characterized by the parameters α and σ. $\alpha(\lambda)$ is the fraction of light which is absorbed per unit path length. $\sigma(\lambda)$ is the fraction of light which has its direction reversed by scattering per unit path length. Both α and σ are functions of wavelength. The model gives rise to simultaneous first order differential equations. These equations can be solved to give expressions for the reflectance and transmission of a colorant layer.

The original Kubelka-Munk theory makes several limiting assumptions. The original theory assumes the boundary condition of diffusely incident light. This is an unrealistic assumption for most real situations. The original K-M theory also assumes that the vehicle containing the colorant particles has an index of refraction equal to that of air. This assumption eliminates the need to consider internal and external reflections at the air-vehicle interface. Unfortunately, this assumption is also not very realistic.

The original Kubelka-Munk theory has been extended by Reichman [19] to eliminate the need for these unrealistic assumptions. Reichman derives an expression for the reflectance of an inhomogeneous material which is valid for collimated light at any angle of incidence. Reichman also uses a method developed by Orchard [18] to take into account both internal and external reflections at the air-vehicle interface.

For our purposes, we consider opaque colorant layers composed of isotropic scatterers. For light incident at an angle θ_l, the extended K-M theory describes the body reflectance R_B as

$$R_B = (1 - R_S)\frac{C(1 - r_i)(R_\infty - D)}{2(1 - r_i R_\infty)cos\theta_l} \qquad (2.7)$$

where R_S is the specular reflectance given by (2.6). r_i is the internal diffuse surface reflectance approximated by Orchard [18] as

$$r_i = 1 - \frac{0.5601 - 0.7099n + 0.3319n^2 - 0.0636n^3}{n^2} \qquad (2.8)$$

where n is the index of refraction of the vehicle. Let $w = \frac{\sigma}{(\alpha+\sigma)}$ be the scattering albedo. R_∞ is the reflectance predicted by original K-M for diffusely incident light and is given by

$$R_\infty = \frac{2 - w - 2\sqrt{1 - w}}{w} \qquad (2.9).$$

C and D result from the solution of Reichman's differential equations and are

$$C = \frac{w\cos\theta_l(2\cos\theta_l + 1)}{1 - 4(1 - w)\cos^2\theta_l} \qquad (2.10)$$

$$D = \frac{2\cos\theta_l - 1}{2\cos\theta_l + 1} \qquad (2.11)$$

One very special case of this model is conservative scattering (also called Lambertian scattering) for which $w(\lambda) = 1$ for all visible wavelengths. A Lambertian scattering model is frequently assumed in computer vision.

2.3. The Reflectance Model

Given our models of both interface and body reflection, we can quantify the reflectance R as

$$R = R_S + R_B \qquad (2.12)$$

where R_S is the Fresnel reflection term and R_B is the Kubelka-Munk body reflection term. The power of the light reflected from a surface at a single wavelength λ_0 towards a viewer is then given by

$$I(\lambda_0) = R(\lambda_0)L(\lambda_0) \qquad (2.13)$$

where $L(\lambda)$ quantifies the power of the incident light.

3. Image Irradiance and Geometry

Image irradiance is a single measure of how much

visible light strikes an area of an image. It is defined as power per unit area. Image irradiance, therefore, contains no direct information about the color of the light striking the image, i.e., many different spectral distributions of light will give the same image irradiance. In this section, we describe the relationship between image irradiance and geometry for both ideal specular surfaces and ideal diffuse surfaces. We note that R_S in (2.12) is defined relative to an ideal specular surface, while R_B in (2.12) is defined relative to an ideal diffuse surface.

3.1. Fresnel Reflection

We begin by analyzing the properties of image irradiance for a perfectly specular surface. In this limiting case, all light incident on the surface will be reflected in such a way that $\theta_v = \theta_l$ and $\overline{L}$, $\overline{N}$, and $\overline{V}$ lie in the same plane (Figure 2).

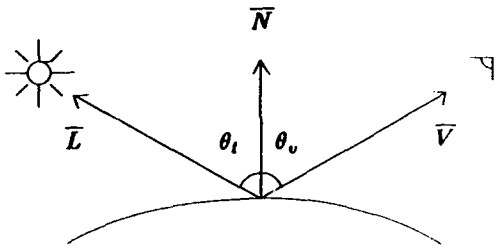


Figure 2. The Reflection Geometry

Thus, all light incident on a surface point from a single direction will be reflected in a single direction.

Consider a small source of radiance L_i and a surface patch dA. Then the power of the light incident on dA is given by

$$\Phi = L_i cos\theta_l dA dw_i \tag{3.1}$$

where dw_i is the solid angle of the source as viewed from dA. If ϕ is taken to be the azimuthal angle, with $\phi = 0$ corresponding to the direction of the projection of $\overline{V}$ onto the tangent plane to the surface, then, by assumption, all of the incident light will be reflected in the direction $\theta_v = \theta_l$, $\phi = 0$. If our imaging system is aligned in this direction, light of power Φ will pass through the lens. For any other position $\overline{V}$, no light from dA will enter the lens and image irradiance resulting from light reflected from dA will be zero. Assuming no losses in the lens, image irradiance E_I will be given by

$$E_I = \frac{\Phi}{dI} = \frac{L_i cos\theta_l dA dw_i}{dI} \tag{3.2}$$

where dI is the image area corresponding to the surface area dA.

The ratio $\frac{dA}{dI}$ is determined by the imaging geometry (Figure 3)

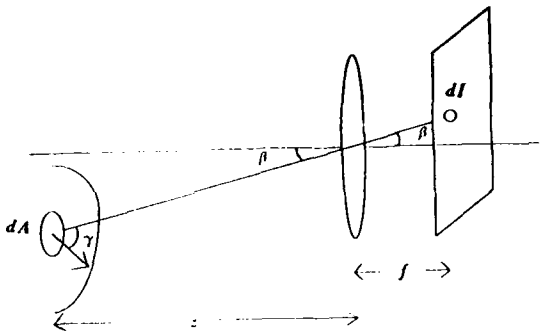


Figure 3. The Imaging Geometry

$$\frac{dA}{dI} = \frac{cos\beta}{cos\gamma}\left(\frac{z}{f}\right)^2 \tag{3.3}$$

giving

$$E_I = \frac{L_i cos\theta_l cos\beta z^2 dw_i}{cos\gamma f^2} \tag{3.4}$$

For most real imaging situations, $\frac{z^2}{f^2}$ will be large and we will measure a large image irradiance for the case of perfect specular reflection.

3.2. Colorant Layer Scattering

A perfectly diffuse (Lambertian) surface reflects all incident light in such a way that the perceived brightness of a surface element is constant with respect to $\overline{V}$. Consider a small source of radiance L_i with solid angle dw_i as viewed from a surface patch. The surface irradiance is then given by

$$E_S = L_i cos\theta_l dw_i \tag{3.5}$$

From our definition of a perfectly diffuse surface, the bidirectional reflectance R must be a constant. Since this surface reflects all incident light, the light reflected into the viewing hemisphere must equal E_S. Therefore, we have

$$\int_{-\pi}^{\pi}\int_{0}^{\frac{\pi}{2}} RL_i cos\theta_l dw_i cos\theta_v sin\theta_v d\theta_v d\phi = L_i cos\theta_l dw_i \tag{3.6}$$

where $sin\theta_v d\theta_v d\phi$ is the solid angle of a patch of size

$d\theta_v$ in zenith angle and $d\phi$ in azimuthal angle. The $cos\theta_v$ occurs in the integrand since we are integrating over projected solid angle.

Equation (3.6) can be solved to give

$$R = \frac{1}{\pi} \tag{3.7}$$

The radiance from the surface is

$$L_S = \frac{1}{\pi} L_i cos\theta_i dw_i. \tag{3.8}$$

The relationship between surface radiance and image irradiance is derived in many places including [12] and is given by

$$E_I = L_S \left(\frac{\pi}{4}\right) \left(\frac{d}{f}\right)^2 cos^4\beta \tag{3.9}$$

where d is the diameter of the lens. For a Lambertian surface we have

$$E_I = \frac{L_i}{4} cos\theta_i dw_i \left(\frac{d}{f}\right)^2 cos^4\beta \tag{3.10}$$

In the usual case for which the image covers a narrow angle, β will be nearly zero and $cos^4\beta$ will be nearly unity.

Here we see an important result of the geometric difference between specular reflection and diffuse reflection. The result is that for almost all imaging situations, the image irradiance corresponding to specular reflection is much larger than the image irradiance corresponding to diffuse reflection. Qualitatively, the reason for this difference is that the specularly reflected light is concentrated in a single direction, while the diffusely reflected light is spread out over the viewing hemisphere. We will return to this result in section 9 when we describe a method which can be used to distinguish specularly reflected light from diffusely reflected light.

4. Color

We define the color of light reflected from a surface to be the light's spectral power distribution $I(\lambda)$ in the visible range. Thus

$$I(\lambda) = L(\lambda)R(\lambda) \tag{4.1}$$

where $L(\lambda)$ is the spectral power distribution of the light incident on the surface and $R(\lambda)$ is the spectral reflectance of the surface. In this section, we describe generic properties of the function $R(\lambda)$ with respect to both geometry and the properties of the reflecting material.

4.1. Fresnel Reflection

4.1.1. Homogeneous Materials

Optically homogeneous materials have a constant index of refraction throughout the material. Light which is incident on a homogeneous material is either specularly reflected at the surface or absorbed by the material. Since light is not scattered in the body of the material, no light which enters the material returns through the surface. Metals provide the most abundant examples of homogeneous materials.

The spectral reflectance of a homogeneous material is determined entirely by the Fresnel component of reflectance R_S. For fixed geometry, the variation of R_S with λ depends on the complex index of refraction $M = n(\lambda) - iK_n(\lambda)$. For some homogeneous materials, M can vary considerably with wavelength. Copper, for example, reflects long visible wavelengths much more efficiently than short visible wavelengths. Conversely, the reflectance of aluminum is approximately constant across the visible spectrum.

4.1.2. Inhomogeneous Materials

The body of an inhomogeneous material is made up of colorant particles embedded in a vehicle. While most homogeneous materials are characterized by a large extinction coefficient K_n, inhomogeneous materials have a negligible extinction coefficient across the visible spectrum ($K_n(\lambda) \approx 0$). For inhomogeneous materials $n(\lambda)$ depends on wavelength, but this dependence is typically small. For most inhomogeneous materials, $n(\lambda)$ is constant to less than five percent across the visible spectrum [14]. Since both n and K_n are nearly constant for visible light, R_S is constant with respect to λ. R_S is a function of only geometry for inhomogeneous materials.

4.1.3. Geometrical Effects

From the Fresnel equations (section 2), we have that for fixed n and fixed K_n the specular reflectance is approximately constant over a large range of incidence angles (roughly $0° \leq \theta_I < 70°$) [22]. As θ_I nears $\pi/2$, however, R_S approaches unity for all values of the complex index of refraction. Consequently, as we approach glancing incidence, the color of the specularly reflected light approaches the color of the incident light.

4.2. Colorant Layer Scattering

4.2.1. Inhomogeneous Materials

The color of an inhomogeneous material is primarily due to the scattering and absorbing characteristics of

colorant layers. These characteristics are described by the parameters $\alpha(\lambda)$ and $\sigma(\lambda)$. In the limiting case of a Lambertian surface, reflectance is constant with respect to wavelength. Thus, a true Lambertian surface will always appear white. On the other hand, the colorant particles in real materials tend to selectively absorb certain wavelengths of light while transmitting others. This selective absorption is the primary cause of the variation of R_B with λ.

4.2.2. Geometrical Effects

We have already observed that the color of a Lambertian surface is constant with respect to geometry. Although (2.7) implies that, in general, the color of light scattered from the body of a material depends on geometry, this dependence is usually small. Figure 4 is a plot of $R_B(\theta_l)$ for different values of the scattering albedo w. The line with $R_B(\theta_l) = 1$ corresponds to the conservative scattering case.

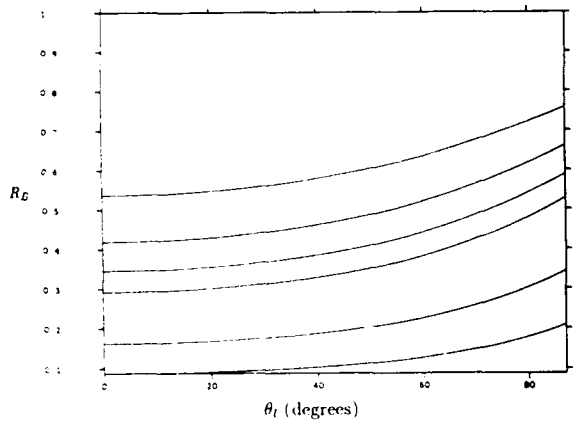


Figure 4. $R_B(\theta_l)$

II. COLOR RECOVERY AND REPRESENTATION

We now begin the second major part of this paper. In the next four sections, we develop methods for the recovery and representation of color.

5. Recovering the Function $I(\lambda)$

In this section, we describe a method for recovering the function $I(\lambda)$ at a point in an image. At each image point, we measure the outputs s_i of n sensors. Each sensor has a certain wavelength sensitivity which we denote by the function $f_i(\lambda)$. Therefore, at each image point we have the measured values s_i $(0 \le i \le n-1)$ given by

$$s_i = \int_\lambda f_i(\lambda) I(\lambda) d\lambda \tag{5.1}$$

where λ ranges over the entire electromagnetic spectrum. For our purposes the filters $f_i(\lambda)$ will be nonzero only for values of λ in the visible range (i.e. 400 nm $\le \lambda \le$ 700 nm).

Suppose we approximate the function $I(\lambda)$ by a linear combination of m basis functions $I_j(\lambda)$. If we let a_j be the components of $I(\lambda)$ on this basis, then we have

$$I(\lambda) \approx \sum_{0 \le j \le m-1} a_j I_j(\lambda) \tag{5.2}.$$

Substituting, (5.1) becomes

$$s_i = \int_\lambda f_i(\lambda) \Big(\sum_{0 \le j \le m-1} a_j I_j(\lambda) \Big) d\lambda \tag{5.3}$$

which may be written

$$s_i = \sum_{0 \le j \le m-1} a_j \Big(\int_\lambda f_i(\lambda) I_j(\lambda) d\lambda \Big) \tag{5.4}$$

Let

$$K_{ij} = \int_\lambda f_i(\lambda) I_j(\lambda) d\lambda \tag{5.5}$$

denote the integral in (5.4). Then K_{ij} is a constant which depends only on the ith filter function and the jth basis function. Let $\bar{s}$ be the n-dimensional vector defined by $\bar{s}(i) = s_i$. Let $\overline{K}$ be the n x m constant matrix defined by $\overline{K}(i,j) = K_{ij}$. Let $\bar{a}$ be the m-dimensional vector defined by $\bar{a}(i) = a_i$. Then we have the linear system of n equations

$$\bar{s} = \overline{K}\bar{a} \tag{5.6}.$$

If we choose our filters $f_i(\lambda)$ and basis functions $I_j(\lambda)$ such that $\overline{K}$ has maximal rank, then the n sensor outputs $s_0, s_1, ..., s_{n-1}$ uniquely determine n components $a_0, a_1, ..., a_{n-1}$ of $I(\lambda)$. Therefore, by letting m=n in (5.6) we can recover an estimate of the function $I(\lambda)$ on the basis $I_j(\lambda)$.

6. Selecting the Sensors $f_i(\lambda)$

Given the recovery technique described in section

5, we examine how a careful choice of the sensors $f_i(\lambda)$ can improve the quality of our recovered approximation to $I(\lambda)$. In section 5, we suggested only that the sensors $f_i(\lambda)$ should be chosen such that $\bar{K}$ has maximal rank. In this section, we derive expressions for the $f_i(\lambda)$ which guarantee we will recover the least error polynomial approximation to $I(\lambda)$.

Since we are considering a polynomial approximation to $I(\lambda)$, we choose our m basis functions $I_j(\lambda)$ to span the space of polynomials of degree less than m. We define the least error polynomial approximation to be the choice of d_i which minimizes

$$\int_{-1}^{1}\left[I(\lambda)-\sum_{0\le i\le m-1} d_i\lambda^i\right]^2 d\lambda \qquad (6.1)$$

where for convenience we have scaled the visible spectrum to be the range $-1 \le \lambda \le 1$.

Any polynomial of degree less than m can be written as a linear combination of the first m Legendre polynomials [5]. Let a_i^* be the coefficients of the least error polynomial on the basis of normalized Legendre polynomials $p_i(\lambda)$ given by

$$p_i(\lambda) = \sqrt{\frac{2i+1}{2}}\,P_i(\lambda) \qquad (6.2)$$

where $P_i(\lambda)$ is the Legendre polynomial of degree i. The functions $p_i(\lambda)$ are normalized in the sense that

$$\int_{-1}^{1}[p_i(\lambda)]^2 d\lambda = 1 \qquad (6.3).$$

The constants a_i^* minimize

$$F = \int_{-1}^{1}\left[I(\lambda)-\sum_{0\le i\le m-1} a_i^* p_i(\lambda)\right]^2 d\lambda \qquad (6.4)$$

Define

$$c_i = \int_{-1}^{1} I(\lambda)p_i(\lambda)d\lambda. \qquad (6.5)$$

Then using (6.5) and the orthogonality of Legendre polynomials, (6.4) may be written

$$F = \int_{-1}^{1}\left[I^2(\lambda)-\sum_{0\le i\le m-1} 2a_i^*c_i+\sum_{0\le i\le m-1}(a_i^*)^2\right]d\lambda \qquad (6.6)$$

$$= \int_{-1}^{1}\left[I^2(\lambda)+\sum_{0\le i\le m-1}(a_i^*-c_i)^2-\sum_{0\le i\le m-1} c_i^2\right]d\lambda \qquad (6.7)$$

From (6.7) we see that the minimizing values of a_i^* are given by $a_i^* = c_i$, $0 \le i \le m-1$.

For an arbitrary choice of functions $f_i(\lambda)$ such that $\bar{K}$ has maximal rank, the procedure of section 5 will not in general recover the minimal error polynomial given by the coefficients a_i^*. But by choosing the functions $f_i(\lambda)$ appropriately, we can guarantee our technique will recover the least error polynomial. One such choice is $f_i(\lambda) = p_i(\lambda)$ $0 \le i \le m-1$. For this case, we will measure $\bar{s} = \bar{c}$. Using (5.6) we will recover the least error approximation $a_i = a_i^*$. One way to see this is to rewrite (5.2) in terms of $p_j(\lambda)$

$$I(\lambda) = \sum_{0\le j\le m-1} a_j I_j(\lambda) = \sum_{0\le j\le m-1} a_j' p_j(\lambda) \qquad (6.8)$$

For this expansion of $I(\lambda)$ in $p_j(\lambda)$ the orthogonality of Legendre polynomials and (5.5) give us

$$K_{ij} = \begin{cases} 1, & \text{if } i = j; \\ 0, & \text{if } i \ne j. \end{cases} \qquad (6.9)$$

so that $\bar{K}$ is the identity matrix. From (5.6) $a = s$ and the recovered polynomial will be the least error polynomial $\sum_{0\le j\le m-1} a_j^* p_j(\lambda)$.

Since for physical sensors we require $f_i(\lambda) \ge 0$ for $1 \le \lambda \le 1$ we cannot directly use $f_i(\lambda) = p_i(\lambda)$ since $p_i(\lambda)$ is negative for some values of i and λ. Below we suggest a method for arriving at the least error polynomial approximation using an equivalent number of physically realizable sensors. We are required to compute the values

$$c_i = \int_{-1}^{1} I(\lambda)p_i(\lambda)d\lambda = \sqrt{\frac{2i+1}{2}}\int_{-1}^{1} I(\lambda)P_i(\lambda)d\lambda \qquad (6.10)$$

For c_0 we have

$$c_0 = \frac{1}{\sqrt{2}}\int_{-1}^{1} I(\lambda)d\lambda \qquad (6.11)$$

which is realizable by using $f_0(\lambda) = 1$. Since $P_i(\lambda) \ge -1$ for $-1 \le \lambda \le 1$ we can compute

$$c_i = \sqrt{\frac{2i+1}{2}}\left[\int_{-1}^{1} I(\lambda)[P_i(\lambda)+1]d\lambda - \sqrt{2}c_0\right] \qquad (6.12)$$

since $P_i(\lambda) + 1 \geq 0$ for $(-1 \leq \lambda \leq 1)$ and c_0 is known from (6.11).

7. A Metric Space for Colors

In this section we develop a metric space for physical colors. There are two important reasons why a vision system should possess a color metric. First, a color metric allows a vision system to determine how closely a perceived color matches a known color. Second, a color metric is necessary if a system hopes to locate color discontinuities in an image. It is well known that the human visual machinery includes a color metric [13]. Our development, however, is motivated by the techniques of functional analysis. No attempt is made to relate our color metric to the color metric implicit in human vision. Nevertheless, it is likely that both metrics serve their respective vision systems in similar ways.

We begin by distinguishing the total power of the signal $I(\lambda)$ from its color. The total power of $I(\lambda)$ is given by

$$E = \int_{-1}^{1} I(\lambda)d\lambda \tag{7.1}$$

where the interval [-1,1] represents the visible spectrum. Total power is a physical analog of the psychological concept of brightness. We define the physical color of $I(\lambda)$ by the function $\widehat{I(\lambda)}$ having unit total power

$$\widehat{I(\lambda)} = \frac{I(\lambda)}{\int_{-1}^{1} I(\lambda)d\lambda} \tag{7.2}$$

Physical color $\widehat{I(\lambda)}$ is related to the psychological concept of hue. The space of physical colors is the space of all continuous nonnegative functions $\widehat{I(\lambda)}$ on [-1,1] having unit total power.

Given any two physical colors $\widehat{I_1(\lambda)}$ and $\widehat{I_2(\lambda)}$ we define the distance from $\widehat{I_1(\lambda)}$ to $\widehat{I_2(\lambda)}$ by

$$d(\widehat{I_1(\lambda)}, \widehat{I_2(\lambda)}) = \sqrt{\int_{-1}^{1} \left[\widehat{I_1(\lambda)} - \widehat{I_2(\lambda)}\right]^2 d\lambda} \tag{7.3}$$

We note that (7.3) satisfies the properties of a distance function, namely

1. $d(\widehat{I_1(\lambda)}, \widehat{I_2(\lambda)}) > 0$ if $\widehat{I_1(\lambda)} \neq \widehat{I_2(\lambda)}$
2. $d(\widehat{I_1(\lambda)}, \widehat{I_1(\lambda)}) = 0$
3. $d(\widehat{I_1(\lambda)}, \widehat{I_2(\lambda)}) = d(\widehat{I_2(\lambda)}, \widehat{I_1(\lambda)})$
4. $d(\widehat{I_1(\lambda)}, \widehat{I_2(\lambda)}) \leq d(\widehat{I_1(\lambda)}, \widehat{I_3(\lambda)}) + d(\widehat{I_3(\lambda)}, \widehat{I_2(\lambda)})$

(1),(2), and (3) easily follow from (7.3) and (4) follows from the Cauchy-Schwartz inequality. Therefore, d is a metric and the space of physical colors under d is a metric space in the topological sense.

For reasons relevant to our application, we use the Euclidean distance of (7.3) rather than the commonly used maximum distance defined by

$$d_{max}(\widehat{I_1(\lambda)}, \widehat{I_2(\lambda)}) = \max_{-1 \leq \lambda \leq 1} |\widehat{I_1(\lambda)} - \widehat{I_2(\lambda)}| \tag{7.4}$$

The distance of (7.3) provides more stability than (7.4) for measured functions $\widehat{I_1(\lambda)}$ and $\widehat{I_2(\lambda)}$. While inaccuracies over a small range of λ can cause large deviations of d_{max}, the distance d of (7.3) depends on the distance between the functions integrated over the entire spectrum. Therefore, the distance d will usually give a more reliable characterization of the distance between two measured physical colors than the distance d_{max}.

8. The Normalized Legendre Polynomial Representation

In section 6, we showed how to select sensors to recover the least error polynomial approximation to the function $I(\lambda)$. The derivation itself suggested that the most convenient representation for this approximation is the basis of normalized Legendre polynomials given by (6.2). Our implementation uses this basis to represent color. In this section, we show how the basis of normalized Legendre polynomials not only facilitates the recovery of the least error approximation to $I(\lambda)$, but also how this basis simplifies many of the computations performed in color space.

8.1. Computing Physical Color $\widehat{I(\lambda)}$

Using the normalized Legendre polynomial representation, the technique of section 5 recovers an approximation to $I(\lambda)$ of the form

$$I(\lambda) = \sum_{0 \leq i \leq m-1} a_i p_i(\lambda) \tag{8.1}$$

The total power of $I(\lambda)$ is given by

$$E = \int_{-1}^{1} I(\lambda)d\lambda = \int_{-1}^{1} \Big[\sum_{0 \leq i \leq m-1} a_i p_i(\lambda)\Big] d\lambda = \sqrt{2}a_0 \tag{8.2}$$

where the last step follows from the orthogonality of the functions $p_i(\lambda)$. Therefore, the total power of $I(\lambda)$ may be determined by only considering the first coefficient

in the normalized Legendre polynomial representation. From the total power of $I(\lambda)$, we can determine the physical color $\widehat{I(\lambda)}$ by

$$\widehat{I(\lambda)} = \frac{I(\lambda)}{\sqrt{2a_0}} = \sum_{0\le i\le m-1} \hat{a}_i p_i(\lambda) \qquad (8.3)$$

where

$$\hat{a}_i = \frac{a_i}{\sqrt{2a_0}} \qquad (8.4)$$

8.2. Metric Space Properties

For two signals $I_1(\lambda)$ and $I_2(\lambda)$ given by

$$I_1(\lambda) = \sum_{0\le i\le m-1} r_i p_i(\lambda), \qquad I_2(\lambda) = \sum_{0\le i\le m-1} s_i p_i(\lambda) \qquad (8.5)$$

the physical colors are

$$\widehat{I_1(\lambda)} = \sum_{0\le i\le m-1} \hat{r}_i p_i(\lambda), \qquad \widehat{I_2(\lambda)} = \sum_{0\le i\le m-1} \hat{s}_i p_i(\lambda) \qquad (8.6)$$

where the $\hat{r}_i$ and $\hat{s}_i$ are computed as in (8.4).

The color space distance between $\widehat{I_1(\lambda)}$ and $\widehat{I_2(\lambda)}$ is given by

$$d(\widehat{I_1(\lambda)}, \widehat{I_2(\lambda)}) = \sqrt{\int_{-1}^{1} \Big[\sum_{0\le i\le m-1} (\hat{r}_i - \hat{s}_i) p_i(\lambda)\Big]^2 d\lambda} \qquad (8.7)$$

which simplifies because of orthogonality to

$$d(\widehat{I_1(\lambda)}, \widehat{I_2(\lambda)}) = \sqrt{\sum_{0\le i\le m-1} (\hat{r}_i - \hat{s}_i)^2} \qquad (8.8)$$

8.3. Color Space as a Subset of R^{m-1}

From (8.3), our representation for physical colors is

$$\widehat{I(\lambda)} = \sum_{0\le i\le m-1} \hat{a}_i p_i(\lambda) \qquad (8.9).$$

From (8.4) we see that all physical colors have $a_0 = 1/\sqrt{2}$. We can write

$$\widehat{I(\lambda)} = \frac{1}{2} + \sum_{1\le i\le m-1} \hat{a}_i p_i(\lambda) \qquad (8.10).$$

We can consider the physical color $\widehat{I(\lambda)}$ to be the point in R^{m-1} with coordinates $\hat{a}_1, \hat{a}_2, \ldots, \hat{a}_{m-1}$. Since we require $\widehat{I(\lambda)} \ge 0$ on $-1 \le \lambda \le 1$, physical colors form a proper subset of R^{m-1}. To be precise, physical colors are those points of R^{m-1} for which

$$\sum_{1\le i\le m-1} \hat{a}_i p_i(\lambda) \ge -\frac{1}{2} \qquad -1 \le \lambda \le 1 \qquad (8.11).$$

Let C^{m-1} be the set of points in R^{m-1} which are physical colors. Points in C^{m-1} will be referred to as

$$a = (\hat{a}_1, \hat{a}_2, \ldots, \hat{a}_{m-1}) \qquad (8.12)$$

where the coordinates $\hat{a}_1, \hat{a}_2, \ldots, \hat{a}_{m-1}$ have the same significance as in (8.10).

An important consequence of viewing colors as points in C^{m-1} is that the usual euclidean metric in R^{m-1} is equivalent to the metric we defined for physical colors in section 7. This is seen by examining (8.8) and recalling that $\hat{r}_0 = \hat{s}_0$. Thus, our representation in terms of normalized Legendre polynomials allows intuition about the familiar distance in R^{m-1} to be applied to distance in color space.

Another useful property of the color space C^{m-1} is that the color corresponding to an arbitrary additive combination of two functions $I_1(\lambda)$ and $I_2(\lambda)$ will lie on the line in C^{m-1} which connects the physical colors C_1 and C_2 corresponding to $I_1(\lambda)$ and $I_2(\lambda)$. Let $I_1(\lambda)$ and $I_2(\lambda)$ be the functions given by

$$I_1(\lambda) = \sum_{0\le i\le m-1} r_i p_i(\lambda), \qquad I_2(\lambda) = \sum_{0\le i\le m-1} s_i p_i(\lambda) \qquad (8.13)$$

The corresponding colors in C^{m-1} are

$$C_1 = (\hat{r}_1, \hat{r}_2, \ldots, \hat{r}_{m-1}), \qquad C_2 = (\hat{s}_1, \hat{s}_2, \ldots, \hat{s}_{m-1}). \qquad (8.14)$$

Consider the linear combination $I_3(\lambda)$ given by

$$I_3(\lambda) = K_1 I_1(\lambda) + K_2 I_2(\lambda) \quad K_1, K_2 \ge 0 \qquad (8.15)$$

Substituting gives

$$I_3(\lambda) = \sum_{0 \le i \le m-1} (K_1 r_i + K_2 s_i) p_i(\lambda) \qquad (8.16).$$

The color of $I_3(\lambda)$ in C^{m-1} is C_3 where

$$C_3 = \left(\frac{K_1 r_1 + K_2 s_1}{\sqrt{2}(K_1 r_0 + K_2 s_0)}, \frac{K_1 r_2 + K_2 s_2}{\sqrt{2}(K_1 r_0 + K_2 s_0)}, \ldots, \frac{K_1 r_{m-1} + K_2 s_{m-1}}{\sqrt{2}(K_1 r_0 + K_2 s_0)} \right) \qquad (8.17).$$

C_3 can be written as

$$C_3 = (\hat{r}_1 + u(\hat{s}_1 - \hat{r}_1), \hat{r}_2 + u(\hat{s}_2 - \hat{r}_2), \ldots, \hat{r}_{m-1} + u(\hat{s}_{m-1} - \hat{r}_{m-1})) \qquad (8.18)$$

where

$$u = \frac{K_2 s_0}{K_1 r_0 + K_2 s_0} \qquad (8.19)$$

Therefore C_3 lies on the line in C^{m-1} connecting C_1 and C_2.

III. COLOR ALGORITHMS

We now move to the third major part of this paper. In the next three sections, we derive algorithms to extract invariant properties of objects from images. In section 12, we present experimental results.

9. Physical Segmentation

Given an image, it is useful to locate the places at which image irradiance is discontinuous. These discontinuities are important because they usually correspond to significant physical events in a scene. Many techniques have been developed to detect image irradiance discontinuities. A summary of some of the issues involved in edge detection, as well as an elegant approach to the problem, is given in [17].

While locating irradiance discontinuities has attracted much attention in computer vision, less work has been done on the important problem of identifying the underlying causes of these discontinuities. The most common physical causes of image irradiance discontinuities are illumination discontinuities, surface orientation discontinuities, specular discontinuities, pigment density discontinuities, and material discontinuities. We will refer to the problem of classifying image irradiance discontinuities according to their physical cause as the physical segmentation problem.

Some progress has been made on classifying certain kinds of irradiance discontinuities. Herskovits and Binford [9] and Horn [11] discuss ways to classify edges in images of polyhedra. Binford [3] examines ways to distinguish discontinuities due to illumination, geometry, and reflectance. Witken [24] attempts to classify an edge by using intensity correlation across the edge. In the context of computing intrinsic images, Barrow and Tenenbaum [1] describe methods for for classifying certain kinds of edges in a limited domain.

Some progress has also been made in using color to classify edges. Rubin and Richards [20] have proposed a method for finding material discontinuities, but their assumptions and physical models appear limiting. Gershon, Jepson, and Tsotsos [6] discuss a more general method for distinguishing material changes from shadow boundaries. Shafer [21] has developed a method to separate diffuse scattering from specular reflection using color. The scope of this method is limited to inhomogeneous materials for which the color of the diffuse reflection differs from that of the specular reflection.

The complete solution to the physical segmentation problem is the subject of another report. Here we restrict ourselves to discussing distinctive properties of the image irradiance discontinuities which occur where the specular component of the reflected light becomes significant.

As discussed in section 3, the most conspicuous feature of specular reflection is that it is invariably associated with image irradiance values which are much larger than those in neighboring image regions. Moreover, the expected magnitude of the difference in irradiance is directly related to properties of the imaging system which are often known. Another quasi-invariant property of specular features is that they are typically small, especially for curved surfaces. Thus unless a specularity occurs at the edge of a surface, it will be surrounded on all sides by diffusely reflected light of approximately the same color and power. Finally, we observe that most specular discontinuities occur in places where the reflecting surface is continuous. It has been shown that image irradiance for diffusely reflected light can be used to compute local descriptions of a surface to at least second order [10]. It has also been shown that image irradiance for specularly reflected light can be used to compute similar local surface descriptions [7]. At specular image irradiance discontinuities, we usually expect the first few derivatives of the surface to be continuous. Thus by comparing the local surface descriptions generated by [10] and [7] we have another way to verify the presence of a specular image irradiance discontinuity.

10. Generic Classification of Materials

In this section and the next, we describe procedures for using color to extract distinctive invariant properties of objects from images. In this section, we show how color can be used to recover a symbolic description of the material an object is made of. In section 11, we describe a method for recovering an object's surface spectral reflectance.

Classifying objects according to material is important because material is an invariant property of an object. It is very valuable, for example, to be able to decide that an object is metal rather than plastic or painted wood rather than dyed cloth.

We showed in section 4 that an important property of a material is whether it is optically homogeneous or optically inhomogeneous. By examining the physics of reflection described in section 2, we have derived a procedure for classifying a material as either homogeneous or inhomogeneous.

Homogeneous materials reflect light only from the surface. The color of this reflected light is determined by the Fresnel equations from the complex index of refraction of the material as a function of wavelength. For a single color of illumination $L(\lambda)$, the color of light reflected from a homogeneous material will be nearly constant with only slight variations due to changing geometry.

Inhomogeneous materials both reflect light from the surface and scatter light from the body of the material. The color of the light reflected from the surface is determined by the index of refraction of the vehicle. The color of the light scattered from the body is determined by the selective absorption properties of the colorant particles embedded in the vehicle. In general, the color of the surface reflected light will be different from the color of the body scattered light. Therefore, given a single color of illumination $L(\lambda)$, there will be two distinct colors of light reflected from an inhomogeneous material.

Using the technique discussed in section 9, we are able to find image irradiance discontinuities corresponding to the places where the power of the specularly reflected light becomes significant. Using our metric for color space (section 7), we can examine whether these image irradiance discontinuities coincide with discontinuities in color space. If a color discontinuity is not detected, there is strong evidence for a homogeneous material. If we do detect a color discontinuity, then the material is probably optically inhomogeneous.

We see that in most situations, it is possible to distinguish homogeneous materials from inhomogeneous materials using techniques in color space. Once the homogeneous-inhomogeneous classification has been made, it is possible to use color to distinguish different homogeneous materials, e.g. aluminum and copper, and to distinguish different inhomogeneous materials, e.g. white plastic and red plastic. Our method to achieve this additional level of classification is based on recovering surface spectral reflectance. We describe our method in the next section.

11. Recovering Surface Spectral Reflectance

Another invariant property of an object which is valuable for recognition is the object's surface spectral reflectance. Unfortunately, surface spectral reflectance is not determined by the spectral distribution of the light reflected by a surface. It is this reflected light which is directly sensed by a vision system. The light which is reflected by a surface is the product of the spectral distribution of the incident light and the spectral reflectance of the surface. To recover the spectral reflectance of a surface, some mechanism must be available to factor out the effects of the incident light. Many experiments have shown that the human vision system is capable of making this computation. This ability of humans to see objects as having a constant color despite varying illumination conditions is called color constancy.

Many theories have been advanced to explain color constancy [2]. Our approach is based on the physics of reflection. Another class of approaches views the task as an underconstrained mathematical problem [16], [25]. These approaches identify the assumptions about incident illumination and surface spectral reflectance which are required to make color constancy possible from a purely computational point of view. In other work, psychologists have suggested that the eye selectively adapts to the color of the ambient light. Experiments have shown that this selective adaptation might be partly responsible for human color constancy [8].

Our method for recovering surface spectral reflectance is applicable to instances of surfaces which are illuminated by the same spectral distribution of light as that which illuminates an inhomogeneous object in the scene. This condition is quite general, and is almost always satisfied in real situations where a small number of different illuminants contribute to the image forming process.

We recall from section 4 that for inhomogeneous materials $K_0(\lambda) = 0$ and $n(\lambda)$ is nearly constant across the visible spectrum. From the Fresnel equations, the specular reflectance of an inhomogeneous material is a constant function of wavelength for fixed geometry. Moreover, the Fresnel equations tell us that the specular reflectance for fixed wavelength is constant with respect to geometry for almost all incidence angles. Therefore, for inhomogeneous objects the Fresnel component of the reflectance can be regarded as constant with respect to both geometry and wavelength.

Given the physical segmentation technique (section 9) and our technique for classifying inhomogeneous materials (section 10), we can recover surface spectral reflectance up to a multiplicative constant. We use the previously described procedures to locate a specular-body reflection boundary on an inhomogeneous object. From (2.12) and (2.13), the measured function $I(\lambda)$ is given by

$$I(\lambda) = [R_S(\lambda) + R_B(\lambda)]\, L(\lambda) \qquad (11.1).$$

On the body reflection side of the boundary, $R_S(\lambda) = 0$ giving

$$I'(\lambda) = R_B(\lambda)L(\lambda) \qquad (11.2)$$

where $R_B(\lambda)$ can be considered to be the same as in (11.1) since $R_B(\lambda)$ changes slowly with respect to geometry. We can solve for $R_S(\lambda)L(\lambda)$ using

$$R_S(\lambda)L(\lambda) = I(\lambda) - I'(\lambda) \qquad (11.3)$$

But $R_S(\lambda)$ is constant with respect to λ and geometry. Let $R_S(\lambda) = k$. Therefore, using (11.3) we can compute $L(\lambda)$ up to the constant k by

$$kL(\lambda) = I(\lambda) - I'(\lambda) \qquad (11.4)$$

From (11.2), $R_B(\lambda)$ can now be computed up to the constant 1/k by

$$\frac{1}{k}R_B(\lambda) = \frac{I'(\lambda)}{kL(\lambda)} \qquad (11.5)$$

For a homogeneous material in the scope of $L(\lambda)$, we have

$$I(\lambda) = R_S(\lambda)L(\lambda) \qquad (11.6)$$

because $R_B(\lambda) = 0$. Thus we can compute $R_S(\lambda)$ to within 1/k by

$$\frac{1}{k}R_S(\lambda) = \frac{I(\lambda)}{kL(\lambda)} \qquad (11.7)$$

Therefore, using (11.5) and (11.7) we can compute the surface spectral reflectance for any surface illuminated by $L(\lambda)$ up to a multiplicative constant. Examples of the performance of this method on real images will be given in section 12.

It should not be surprising that there is a fundamental ambiguity in the computed spectral reflectance corresponding to the constant k. From (4.1)

$$I(\lambda) = L(\lambda)R(\lambda) \qquad (11.8).$$

We see that an arbitrary constant t can be introduced into both $L(\lambda)$ and $R(\lambda)$ such that the resultant $I(\lambda)$ will be indistinguishable from $I(\lambda)$ in (11.8):

$$I(\lambda) = \left(\frac{L(\lambda)}{t}\right)(tR(\lambda)) \qquad (11.9)$$

Thus without using additional assumptions, we cannot expect to determine $R(\lambda)$ better than to within a multiplicative constant. We note that physical color (section 7) is independent of these scaling constants.

12. Experimental Results

A simple laboratory setup has been used to test our color methods. We digitize color images using a solid-state camera and four WRATTEN gelatin filters. The camera is equipped with an infrared cutoff filter. Figure 5 illustrates the spectral characteristics of our sensors.

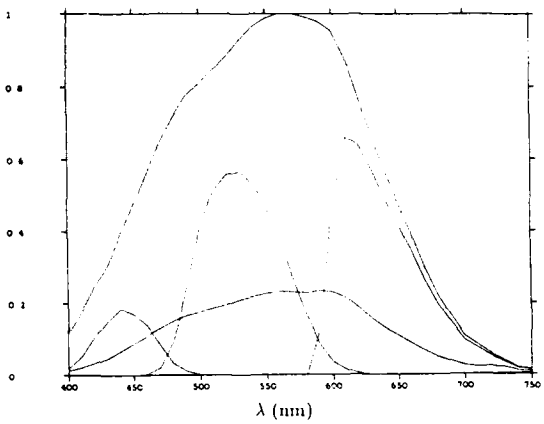


Figure 5. The camera and filter transmission functions

The large amplitude curve indicates the spectral sensitivity of our camera. The four smaller curves are the spectral transmission functions of our filters multiplied by the spectral sensitivity of the camera. These four smaller curves correspond to the functions $f_0(\lambda)$, $f_1(\lambda)$, $f_2(\lambda)$, and $f_3(\lambda)$ of section 5.

Two different light sources have been used for experiments. One is a tungsten halogen lamp of color temperature 3400°K which is typical of indoor illumination. The other lamp has color temperature 4800° and is intended to simulate daylight.

Several simple objects have been used to test our color methods. Our objects include plastic cups, metal cylinders, and painted wooden blocks.

In Figures 6-7, we show the performance of our algorithms on color images of plastic cups illuminated by the $4800°K$ color temperature lamp. Since the image irradiance corresponding to the specular reflection is markedly larger than the image irradiance corresponding to the diffuse reflection, the physical segmentation process of section 9 easily locates the specular-diffuse boundaries. The method of section 5 is then used to recover the function $I(\lambda)$ for both the specularly reflected light and the diffusely reflected light. Figure 6(a) is $I(\lambda)$ for the Fresnel reflection from a blue cup. It agrees well with the actual color of the light source. Figure 6(b) is $I(\lambda)$ for the diffuse reflection from the blue cup. From the large color difference between Figure 6(a) and Figure 6(b), the algorithm of section 10 easily is able to identify the plastic cup as being made of an inhomogeneous material. We remark that it would be very difficult to infer that the cup is blue by simply inspecting the color of the reflected light $I(\lambda)$ in Figure 6(b). In fact, the largest amount of power is in the red part of the visible spectrum (near 700 nm). To determine the color of the cup (as distinct from the color of the light reflected from the cup), we must compute the surface spectral reflectance. This is done using the method of section 11. Figure 6(c) shows the spectral reflectance computed for the blue cup. From Figure 6(c), we can tell that the cup is blue. Figures 7(a), 7(b), and 7(c) show the performance of our algorithms on a color image of a red plastic cup. We see that our algorithms are able to correctly determine the object's material (from Figures 7(a) and 7(b)) and spectral reflectance (Figure 7(c)).

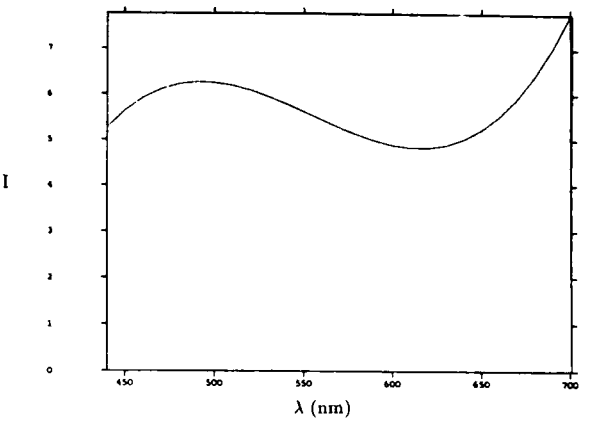


Figure 6(b). Diffuse Reflection from blue cup

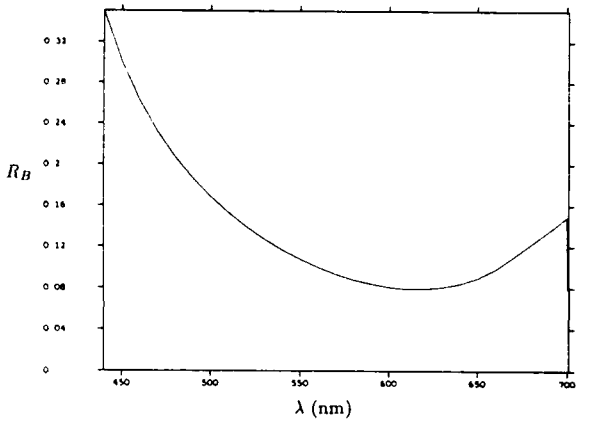


Figure 6(c). Computed Reflectance for blue cup

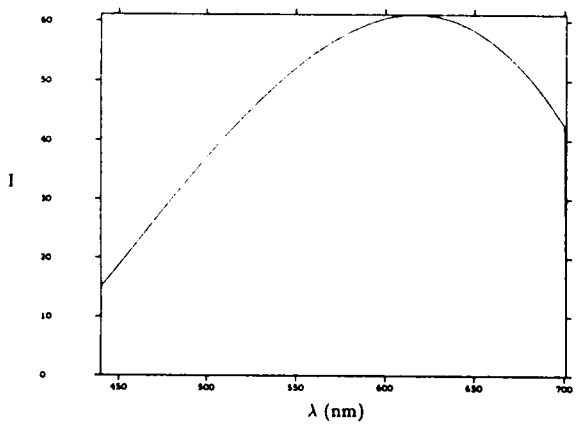


Figure 6(a). Fresnel Reflection from blue cup

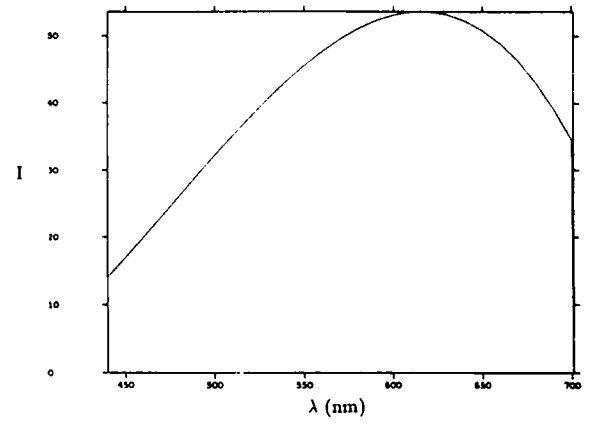


Figure 7(a). Fresnel Reflection from red cup

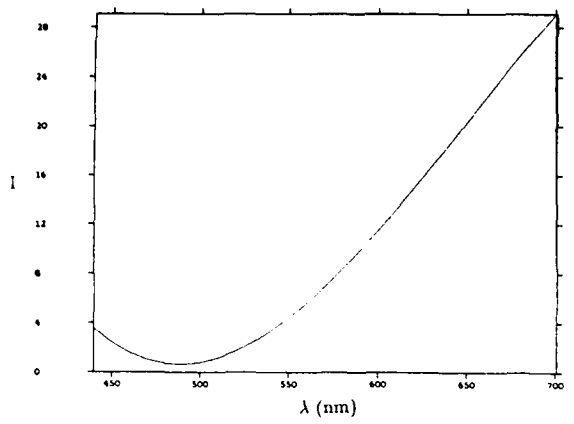


Figure 7(b). Diffuse Reflection from red cup

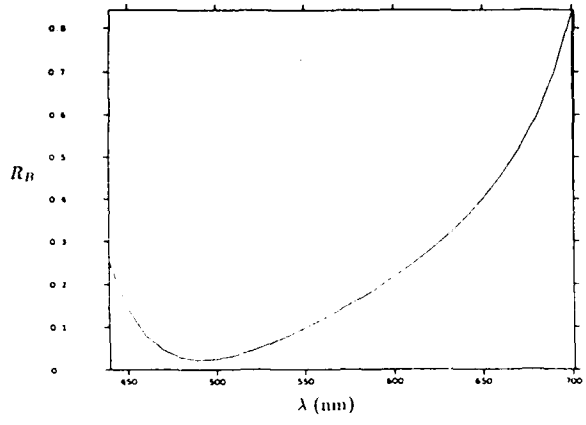


Figure 7(c). Computed Reflectance for red cup

13. Conclusions

In this paper, we have analyzed the importance of color understanding in a general vision system. Starting from general physical models, we have shown that color can play an important role in both the classification of materials and in the recognition of objects. An appropriate use of color, therefore, can significantly extend the capabilities of a vision system.

We have developed two color algorithms which extract invariant properties of objects from images. The first algorithm classifies objects according to material. The second algorithm recovers an object's surface spectral reflectance. Both algorithms have been implemented and consistently produce correct results on real images.

Our color algorithms are part of a general color understanding system. In constructing this system, we have addressed the problems of color recovery and color representation. The input to the system is images captured using sensors with different spectral responses. A robust technique has been developed to recover physical color from these images. We represent physical color by points in a metric space. As we have shown, the properties of our representation greatly facilitate using color to achieve the goals of a general vision system.

Acknowledgements

This work has been supported by an NSF graduate fellowship, AFOSR contract F33615-85-C-5106, and ARPA contract N000-39-84-C-0211.

References

[1] Barrow, H.G., and Tenenbaum, J.M., "Recovering Intrinsic Scene Characteristics from Images", in A.R. Hanson and E.M. Riseman (Eds.), *Computer Vision Systems*. Academic Press, New York, 1978, 3-26.

[2] Beck, J., *Surface Color Perception*, Cornell U. Press, Ithaca, N.Y., 1972.

[3] Binford, T.O., "Inferring Surfaces from Images," *Artificial Intelligence* 17, 1981, 205-245.

[4] Born, M. and Wolf, E., *Principles of Optics*, Pergamon Press, New York, 1959.

[5] Courant, R. and Hilbert, D., *Methods of Mathematical Physics*, Volume 1, Wiley & Sons, New York, 1953.

[6] Gershon, R. and Jepson, A. and Tsotsos, J., " Ambient Illumination and the Determination of Material Changes", *Journal of the Optical Society of America A*, 3(10), 1986, 1700-1707.

[7] Healey, G. and Binford, T.O., "Local Shape from Specularity", Proceedings of ARPA Image Understanding Workshop, USC, 1987.

[8] Helson, H., *Adaptation-Level Theory*, Harper and Row, New York, 1964.

[9] Herskovits, A. and Binford, T.O., "On Boundary Detection", MIT AI Memo 183, 1970.

[10] Horn, B.K.P., "Obtaining Shape from Shading Information," in *The Psychology of Computer Vision*, (P. Winston, Ed.), McGraw-Hill, New York, 1975.

[11] Horn, B.K.P., "Understanding Image Intensities," *Artificial Intelligence* 8, 1977, 201-231.

[12] Horn, B.K.P. and Sjoberg, R., "Calculating the Reflectance Map," *Applied Optics*, Vol. 18, No. 11, June 1979, 1770-1779.

[13] Judd, D. and Wyszecki, G., *Color in Business, Science, and Industry*, Wiley & Sons, New York, 1975.

[14] Kanthack, R., *Tables of Refractive Indices*, Vol. II., App. III, Hilger, London, 1921.

[15] Kubelka, P. and Munk, F., "Ein Beitrag sur Optik der Farbanstriche", *Z. tech. Physik.*, 12, 1931, 593.

[16] Maloney, L. and Wandell, B., "Color Constancy: a method for recovering surface spectral reflectance," *Journal of the Optical Society of America*, Vol. 3, January 1986, 29-33.

[17] Nalwa, V. and Binford, T.O., "On Detecting Edges," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Volume 8, No. 6, November 1986, 699-714.

[18] Orchard, S., "Reflection and Transmission of Light by Diffusing Suspensions," *Journal of the Optical Society of America* Volume 59, Number 12, December 1969, 1584-1597.

[19] Reichman, J., "Determination of Absorption and Scattering Coefficients for Nonhomogeneous Media. 1: Theory," *Applied Optics*, Vol. 12, No. 8, August 1973, 1811-1815.

[20] Rubin, J. and Richards, W., "Color Vision and Image Intensities: When are Changes Material?", MIT AI Memo 631, May 1981.

[21] Shafer, S., "Using Color to Separate Reflection Components", University of Rochester TR 136, April 1984.

[22] Sparrow, E. and Cess, R., *Radiation Heat Transfer*, McGraw-Hill, New York, 1978.

[23] Torrance, K. and Sparrow, E., "Theory for Off-Specular Reflection from Roughened Surfaces,", *Journal of the Optical Society of America*, 57 (1967), 1105-1114.

[24] Witken, A., "Intensity-based Edge Classification," Proceedings of AAAI, 1982, 36-41.

[25] Yuille, A., "A Method for Computing Spectral Reflectance", MIT AI Memo 752, December 1984.

Using a Color Reflection Model to Separate Highlights from Object Color[1]

Gudrun J. Klinker, Steven A. Shafer, and Takeo Kanade

Department of Computer Science, Carnegie-Mellon, Pittsburgh, PA 15213, USA

Abstract

Current methods for image segmentation are confused by artifacts such as highlights, because they are not based on any physical model of these phenomena. In this paper, we present an approach to color image understanding that accounts for color variations due to highlights and shading. Based on the physics of reflection by dielectric materials, such as plastic, we show that the color of every pixel from an object can be described as a linear combination of the object color and the highlight color. According to this model, all color pixels from one object form a planar cluster in the color space whose shape is determined by the object and highlight colors and by the object shape and illumination geometry. We present a method which exploits the color difference between object color and highlight color, as exhibited in the cluster shape, to separate the color of every pixel into a matte component and a highlight component. This generates two intrinsic images, one showing the scene without highlights, and the other one showing only the highlights. The intrinsic images may be a useful tool for a variety of algorithms in computer vision that cannot detect or analyze highlights, such as stereo vision, motion analysis, shape from shading, and shape from highlights. We have applied this method to real images, in a laboratory environment, and we show these results and discuss some of the pragmatic issues endemic to precision color imaging.

1. Introduction

When we look at an image, we can interpret what we see as a collection of shiny and matte surfaces, smooth and rough, interacting with light, shape, and shadow. However, computer vision has not yet been successful at deriving a similar description of surface and illumination properties from an image. The key reason for this failure has been a lack of models or descriptions rich enough to relate pixels and pixel-aggregate properties to these scene characteristics. In the past, most work with color images has considered object color to be a constant property of an object. Color variation on an object was attributed to noise. However, in real scenes, color variation depends to a much larger degree on the optical reflection properties of the scene, which cause the perception of object color, highlights, shadows and shading [3, 9]. As a consequence, color variation cannot be regarded to be of a merely statistical nature. On the contrary, it exhibits characteristics that can be determined and used for color vision.

[1]This material is based upon work supported by the National Science Foundation under Grant DCR-8419990 and by the Defense Advanced Research Projects Agency (DOD), ARPA Order No. 4976, monitored by the Air Force Avionics Laboratory under contract F33615-84-K-1520. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the National Science Foundation or the Defense Advanced Research Projects Agency or the US Government.

This paper presents an approach to color image understanding that accounts for color variations due to highlights and shading. We use a reflection model which describes the color of every pixel from an object as a linear combination of the object color and the highlight color [10]. According to our model, all color pixels from one object form a planar cluster in the color space. The cluster shape is determined by the object and highlight colors and by the object shape and illumination geometry.

We present a method that exploits the color difference between object color and highlight color, as exhibited in the cluster shape, to separate the color of every pixel into a matte component and a highlight component. Our method generates two intrinsic images, one showing the scene without highlights, and the other one showing only the highlights. These intrinsic images can be a useful tool for a variety of algorithms in computer vision that cannot detect or analyze highlights, such as stereo vision, motion analysis, shape from shading and shape from highlights [2, 3, 11]. We demonstrate the applicability of our method to real color images.

We begin by introducing the dichromatic reflection model which describes the interaction of light with opaque dielectric materials. Using this model, we describe the color variation on an object as a function of the reflection properties of the materials, object shapes and sensor characteristics. Next, we demonstrate how our reflection model can be used for the analysis of color images. We present methods to determine the illumination color and to detect and remove highlights from real color images. Finally, we discuss the implications and the future direction of our work.

2. The dichromatic reflection model

When we look at a glossy object, we usually see the reflected light as composed of two colors that typify the highlight areas and the matte object parts. The *dichromatic reflection model* describes this phenomenon for scene configurations in which a single light source of an arbitrary color illuminates opaque dielectric materials [10]. As we will show later in this paper, this model accounts well for real color data under suitable conditions.

When light hits an object of opaque dielectric material, the material interface immediately reflects some percentage of the light, according to Fresnel's law of reflection. In general, the reflected light has approximately the same color as the light source. The remaining percentage of the incident light penetrates into the material body, which then scatters the light and absorbs it at some wavelengths, before it reemits the rest [4, 6, 12]. The color of this light is determined by the illumination color and the reflection properties of the material. Common names for the reflection process at the interface are the terms *specular reflection*, *highlight* or *gloss*, whereas the reflection process in the material body is generally called *diffuse reflection* or *matte color*. We refer to the

two processes as *interface* and *body reflection* because those terms make a more precise, physical distinction between the reflection processes than the geometry-based terms specular and diffuse reflection. We will use this terminology and the corresponding terms highlights and matte object color throughout the paper.

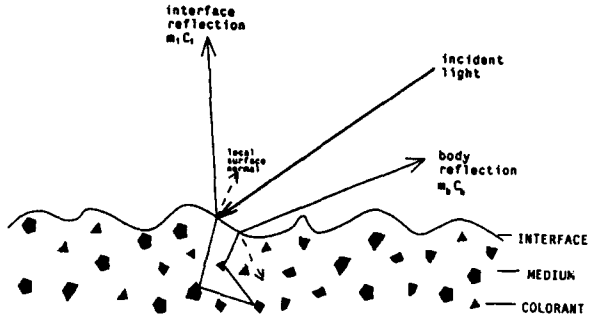


Figure 2-1: Light reflection at dielectric materials

Based on the above discussion, the dichromatic reflection model describes the spectral radiance $L(\lambda,i,e,g)$[2] of a point in the scene as a sum of an interface reflection component $L_i(\lambda,i,e,g)$ and a body reflection component $L_b(\lambda,i,e,g)$. The model assumes that the spectral properties of illumination and reflection on an object are independent of the orientation of the surface, which is a reasonable approximation [10]. We thus decompose each of the two reflection components into a spectral *composition* $c(\lambda)$ that describes a color, and a *magnitude*, $m(i,e,g) \in [0,1]$, that describes a geometric scale factor:

$$L(\lambda,i,e,g) = m_i(i,e,g)\, c_i(\lambda) + m_b(i,e,g)\, c_b(\lambda) \qquad (2.1)$$

When a color TV camera records an image of a scene, it generally uses three color primaries to represent the spectrum of the light that is reflected from the objects towards the camera. In the image formation process, the camera transforms the light of the incoming ray at pixel position (x,y) via tristimulus integration from an infinite light spectrum into a triple of color values, $\mathbf{C}(x,y) = [r,g,b]$. This process sums the amount of light at each wavelength weighted by the transmittance of the color filters and the responsivity of the camera at each wavelength. Because this is a linear transformation, and because the photometric angles, i, e, and g depend on x and y, the dichromatic reflection model can be applied to color pixel values. This allows us to describe the color pixel value $\mathbf{C}(x,y)$ as a linear combination of the vectors representing the colors of interface reflection $\mathbf{C}_i$ and body reflection $\mathbf{C}_b$ at the corresponding point in the scene: point:

$$\mathbf{C}(x,y) = m_i(i,e,g)\, \mathbf{C}_i + m_b(i,e,g)\, \mathbf{C}_b \qquad (2.2)$$

In this equation, the color vectors $\mathbf{C}_i$ and $\mathbf{C}_b$ are constant for a surface, and the scale factors m_i and m_b vary at each pixel.

[2] i, e, and g describe the angles of light incidence and exitance and the phase angle, λ is the wavelength parameter

3. Color variation and object shape

In the previous section we have shown how the pixel color $\mathbf{C}(x,y)$ depends on the optical properties of the scene. We will now discuss the relationship between the colors of all pixels on an object. As a means to model the color variation over an entire object, we use a color histogram in the color space, which is the projection of the colors of all pixels from the object into the color space.

The dichromatic reflection model assumes that there is a single light source in the scene, without ambient light or inter-reflection between objects. Under this assumption, the colors of all pixels from an object are linear combinations of the same interface and body reflection colors $\mathbf{C}_i$ and $\mathbf{C}_b$. Color variation within an object area thus depends only on the geometric scale factors m_i and m_b while $\mathbf{C}_i$ and $\mathbf{C}_b$ are constant. Accordingly, $\mathbf{C}_i$ and $\mathbf{C}_b$ span a *dichromatic plane* in the color space, and the colors of all pixels from one object lie in this plane.

Within the dichromatic plane, the color pixels form a dense cluster. There exists a close relationship between the shape of such a color cluster and the geometric properties of body and interface reflection and the shape of the object. We can use this relationship to determine characteristic features of the color clusters. As an aid to intuition, we will assume that body reflection is approximately Lambertian and that interface reflection is describable by a function with a sharp peak around the angle of perfect mirror reflection. This is a simplified view of the reflection processes in real scenes. However, as we will demonstrate, it is sufficient for our analysis of color images.

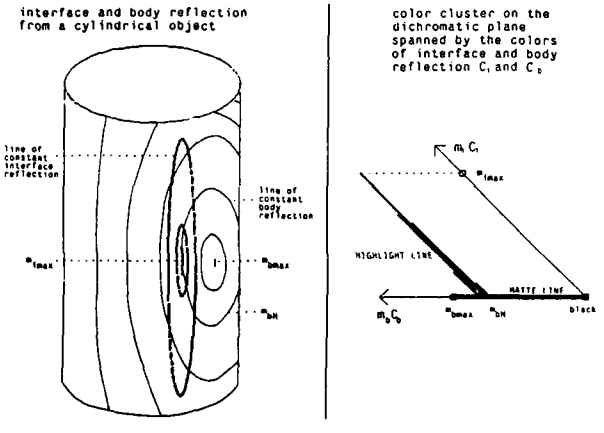


Figure 3-1: The shape of the color cluster for a cylindrical object

Figure 3-1 shows a sketch of a shiny cylinder. The left part of the figure displays the magnitudes of the body and interface components as curves showing the loci of constant body or interface reflection. The right part of the figure shows the corresponding color cluster in the dichromatic plane. This represents the configuration observed in a histogram of pixel values in the color space. To relate the terminology of the dichromatic reflection model to the shape of the color clusters, we classify the color pixels as *matte pixels*, *highlight pixels* or *clipped color pixels*. The following paragraphs discuss the characteristic features of each of these classes.

Matte pixels are projections of points in the scene that exhibit only body reflection in the direction of the viewer. The color of such pixels is thus determined by the color of body reflection, scaled according to the geometrical relationship between the local surface normal of the object and the viewing and illumination directions. Consequently, the colors of the matte pixels form a *matte line* in the color space, in the direction of the body reflection vector $\mathbf{C}_b$.

Highlight pixels are projections of scene points that exhibit both body reflection and interface reflection in the viewing direction. The colors of all pixels in a highlight area that lie on a line of constant body reflection vary only in their respective amounts of interface reflection. The colors of these pixels thus form a straight *highlight line* in the color space that starts at the matte cluster at the position that is determined by their body reflection component m_{bH}. The direction of the highlight line is determined by the interface reflection vector $\mathbf{C}_i$. Combined with the neighboring highlight pixels of slightly different amounts of body reflection, all highlight pixels form a highlight cluster in the color space that has the shape of a skewed wedge. If more than one highlight exists on an object, each of them describes a highlight line in the color space (see Figure 3-2).

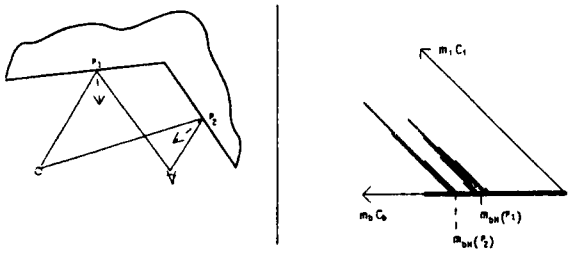


Figure 3-2: Color cluster shapes under changing illumination geometry

The combined color cluster of matte and highlight pixels thus looks like a skewed T or comb. The exact shape of the cluster depends on the illumination geometry. If the phase angle g between the illumination and viewing direction at a highlight is very small, the incidence direction of the light is very close to the surface normal. According to Lambert's law, the body reflection component is maximal at such object points and thus, the starting point m_{bH} for the highlight line is at the tip of the matte line. The wider g becomes at a highlight, the smaller is the amount of underlying body reflection and, thus, the greater is the distance between the tip of the matte line and the starting point m_{bH} of the highlight line (see Figure 3-2). At the same time, small positional variations on the object (as for neighboring pixels) become less influential on the incidence and exitance angles i and e. The highlight thus becomes dimmer and spreads out over a larger area, covering a larger range of values of underlying body reflection. As a result, the highlight line grows wider, exhibiting more strongly the shape of a wedge.

Clipped color pixels are highlight pixels at which the light reflection exceeds the dynamic range of the camera. Depending on the color of the object, the dynamic range may be exceeded at some points in one color band but not in the other two, and the highlight cluster then bends near the wall of the color cube that describes the limit of sensitivity of that color band. A second color band may saturate at some brighter object points, causing the color cluster to bend again at the edge of the limiting walls of both color bands. At the innermost points of the highlight the dynamic range of the camera may even be exceeded in all three color bands, thus looking white, even though the color of illumination may not be white.

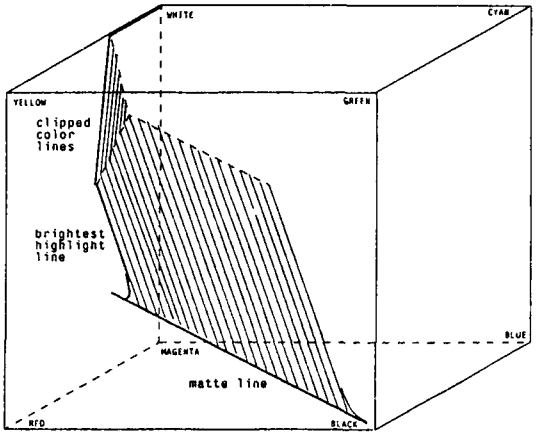


Figure 3-3: Shape of color clusters in color space

Summarizing this discussion, the general shape of a color cluster is displayed in Figure 3-3. Although the color cluster of a specific object may not fill the entire parallelogram, it reliably exhibits some features that can thus be used and searched for by algorithms. *Figure 3-3 displays* these features as bold lines. Color clusters generally provide a *matte line* on which all matte pixels of an object area lie. Depending on the object shape and the illumination geometry, color clusters have some number of highlight lines. We use the *brightest highlight line* as a representative of all these lines. Further lines connected to the highlight lines are *clipped color lines*. An algorithm can analyze a color cluster by searching for these lines in the color space. They determine the general shape of the parallelogram and the orientation of the dichromatic plane.

4. Analysis of real color images

There exist some inherent assumptions in the dichromatic reflection model [10]. However, as we will illustrate, the model accounts well for real color data under suitable conditions. We also present methods for color image analysis that use the shape of the color clusters.

4.1 Color clusters from real color images

We have taken a series of color images in the Calibrated Imaging Laboratory at Carnegie-Mellon. The scene consists of an orange, a green and a yellow plastic cup under white or yellow illumination. Black curtains on the walls were used to eliminate ambient light in the visible spectrum of the scene. We use a spectral linearization method to compensate for the non-linear response of our camera to image brightness. The upper left box of Figure 4-1 shows the linearized image of the orange, the yellow and the green cup under yellow illumination.

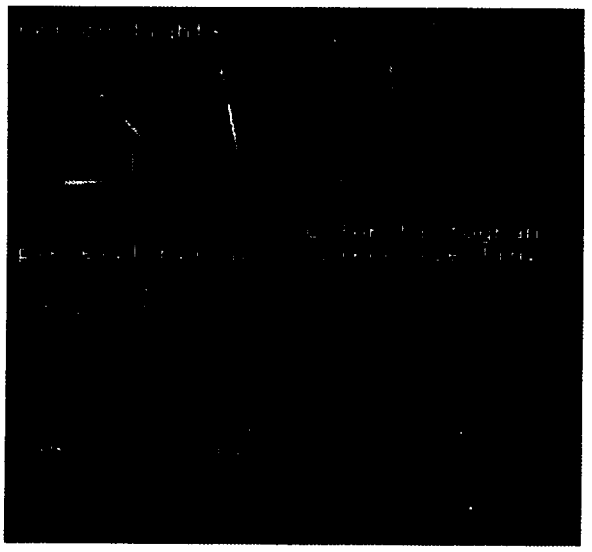


Figure 4-1: Color clusters and dichromatic planes of cups under yellow light

We use a graphical display program which takes a color picture as input and displays the colors of all pixels of selected object areas as color clusters in the color space. The upper right box of Figure 4-1 displays the color histogram of the pixels from the marked areas of the color image (in the upper left box) in the color space. The color space is shown as a color cube, with each dimension of the cube representing the intensity scale of one of the three color primaries. Its origin is at the black corner of the cube.

The color clusters of the cups each lie approximately in dichromatic planes. Within the planes, they form matte and highlight lines, thus demonstrating that real color data follows the theory of the dichromatic reflection model. The color clusters also have clipping lines, due to the bright intensity of the yellow illumination, as reflected from the middle of the highlights. Note that, accordingly, the colors in the middle of the highlight areas look white, whereas the pixels closer to the highlight boundaries are yellow.

4.2 Determining the color of illumination

If several glossy objects of different color are illuminated by the same light source, each object produces a dichromatic plane. Because all of these dichromatic planes contain the same interface reflection vector C_i, they intersect along a single line which is the color of the illumination (see the lower left box of Figure 4-1). This fact can be used by color constancy algorithms [1, 7] that try to remove the influence of the illumination color from the body reflection component, thus "normalizing" the image to a standard white illumination.

4.3 Detecting and removing highlights

We have developed and implemented an algorithm that uses the shape of the color clusters to detect and remove highlights from color images. The program projects the pixels of selected image areas into the color space and fits a dichromatic plane to the color data from each image area. The program then searches within each dichromatic plane for the matte line, the brightest highlight line and lines of clipped colors. These lines are extracted from the dichromatic plane by using a recursive line splitting algorithm [8], and classified as the *matte vector*, the *highlight vector* and *clipped color vectors*. The program assumes that the line starting closest to the black corner is the matte vector. The next one connected to it is classified as the highlight vector. The remaining lines are assumed to describe clipped color data.

Each color pixel of the image is then broken up into its reflection components. In order to remove interface reflection, the program projects the color of every pixel onto the respective dichromatic plane. If the projected color is close to a clipped color vector, it is replaced by the color at the end of the highlight vector. The algorithm then projects the color of every pixel along the highlight vector onto the matte line. The result is the intrinsic *matte image* of the scene. Conversely, the program forms the intrinsic *highlight image* of a scene by projecting the color of every pixel along the matte vector onto a line that is parallel to the highlight vector but goes through the origin of the color space.

The program has been applied to the pictures of an orange, a yellow or a green plastic cup under white or yellow light. Figure 4-2 shows the results we obtained from running the algorithm on an image of the orange cup under white light. The upper boxes show the image of the cup and the color cluster that is generated by the pixels from the marked object area. The lower two boxes display the resulting intrinsic images of our algorithm. The left box shows only the highlight, and the right box shows the object without the highlight. The results of applying the algorithm to the other pictures are similar.

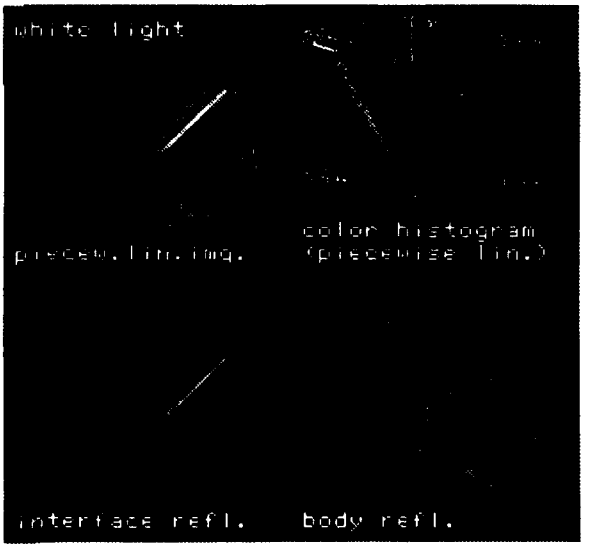


Figure 4-2: Intrinsic images of the orange cup under white illumination

A qualitative inspection of the intrinsic images reveals that our algorithm is able to separate the highlights from the body reflection component. Note that the highlight image displays both the highlight from the middle of the cup and the small amount of gloss that is reflected from the handle of the cup. However, the current algorithm does not yet detect the interface reflection color reliably. The scene of Figure 4-2 was illuminated by white light. In the color cube, the white light corresponds to the diagonal direction, from the black corner to the white corner. Since the highlight line of this

image starts with an already high amount of mainly red body reflection, the diagonal direction of white light is hard to detect and, *in this image, the clipping vector could not be distinguished from* the highlight vector. For this reason, the computed highlight vector misses the red component and the highlight image looks cyan. The algorithm can be improved by using the intersection of several dichromatic planes as an indication of the highlight vector (see figure 4-1).

The potential utility or the intrinsic images stems from two facts: Both the matte and the highlight image have simpler geometric properties than does intensity in a monochrome image, that represents a weighted sum of the two; and the matte image is relatively insensitive to changes in viewpoint from one image to the next.

5. Conclusion

In this paper, we have demonstrated that it is possible to analyze real color images by using a color reflection model. Our model accounts for highlight reflection and matte shading, as well as for the limited dynamic range of cameras. By developing a physical description of color variation in color images, we have developed a method to separate highlight reflection from matte object reflection. The resulting two intrinsic images are promising for improving the results of many other computer vision algorithms that cannot detect or analyze highlights. To demonstrate this, we plan to combine our approach with a photometric stereo system to guide a robot arm [5].

The key point leading to the success of this work is our modeling of highlights as a linear combination of both body and interface reflection. In contrast, previous work on highlight detection in images has generally assumed that the color of the pixels within a highlight is completely unrelated to the object color. This assumption would result in two unconnected clusters in the color space: one line or ellipsoid representing the object color and one point or sphere representing the highlight color. Our model and our color histograms demonstrate that, in real scenes, a transition area exists on the objects from purely matte areas to the spot that is generally considered to be the highlight. This transition area determines the characteristic shapes of the color clusters which is the information that we use to detect and remove highlights. This view of highlights should open the way for quantitative shape-from-gloss analysis, as opposed to the current binary methods based on thresholding intensity.

Our approach to color image analysis may influence research in other areas of color computer vision. The color histograms demonstrate that all color pixels from one object (material) form a *single cluster in the color space*. A color cluster thus groups pixels into areas of constant material properties, independently of illumination geometry influences, such as shading or highlights. This finding can be used by color segmentation algorithms to distinguish material changes from shading or highlight boundaries. Furthermore, the shape of the color clusters reveals some information about the illumination geometry and the object shapes. We are currently investigating these implications of the model to improve color image understanding methods, and we are also considering extensions of the model to account for other material types and more complex illumination conditions including ambient light, light sources in several colors, shadow casting and inter-reflection between objects.

Although the current method has only been applied in a laboratory setting, its initial success shows the value of modeling the physical nature of the visual environment. Our work and the work of others in this area may lead to methods that will free computer vision from its current dependence on signal-based methods for image segmentation.

Acknowledgments

We would like to thank Ruth Johnston-Feller for her comments and suggestions and Georg Klinker and Richard Szeliski for commenting previous drafts of this paper.

References

[1] M. D'Zmura and P. Lennie.
Mechanisms of color constancy.
Journal of the Optical Society of America A (JOSA-A) 3(10):1662-1672, October, 1986.

[2] L. Dreschler and H.-H. Nagel.
Volumetric Model and 3D Trajectory of a Moving Car Derived from Monocular TV Frame Sequences of a Street Scene.
Computer Graphics and Image Processing 20:199-228, 1982.

[3] B.K.P. Horn.
Understanding Image Intensities.
Artificial Intelligence 8(11):201-231, 1977.

[4] R.S. Hunter.
The Measurement of Appearance.
John Wiley and Sons, New York, 1975.

[5] K. Ikeuchi.
Generating an Interpretation Tree From a CAD Model to Represent Object Configurations For Bin-Picking Tasks.
Technical Report CMU-CS-86-144, Department of Computer Science, Carnegie-Mellon University, Pittsburgh, PA 15213, August, 1986.

[6] D.B. Judd and G. Wyszecki.
Color in Business, Science and Industry.
John Wiley and Sons, New York, 1975.

[7] H.-C. Lee.
Method for computing the scene-illuminant chromaticity from specular highlights.
Journal of the Optical Society of America A (JOSA-A) 3(10):1694-1699, October, 1986.

[8] T. Pavlidis.
Structural Pattern Recognition.
Springer Verlag, Berlin, Heidelberg, New York, 1977.

[9] J.M. Rubin and W.A. Richards.
Color Vision and Image Intensities: When are Changes Material?
Biological Cybernetics 45:215-226, 1982.

[10] S.A. Shafer.
Using Color to Separate Reflection Components.
COLOR research and application 10(4):210-218, Winter, 1985.
Also available as technical report TR 136, University of Rochester, NY, April 1984.

[11] C.E. Thorpe.
FIDO: Vision and Navigation for a Robot Rover.
PhD thesis, Computer Science Department, Carnegie-Mellon University, December, 1984.
available as technical report CMU-CS-84-168.

[12] S.J. Williamson and H.Z. Cummins.
Light and Color in Nature and Art.
John Wiley and Sons, New York, 1983.

RECOGNIZING UNEXPECTED OBJECTS: A PROPOSED APPROACH

Azriel Rosenfeld

Center for Automation Research
University of Maryland
College Park, MD 20742

ABSTRACT

Humans can recognize familiar, but unexpected objects, belonging to highly variable object classes, in a fraction of a second—a few hundred cycles of the neural visual "hardware". This position paper suggests computational techniques that could serve as a basis for this type of object recognition if implemented on appropriate parallel hardware.

1. INTRODUCTION

Imagine that you are viewing a random sequence of slides, each showing a single familiar object on a blank background. The objects might include specific kinds of animals, plants, furniture, household implements, tools, office equipment, alphanumeric characters; there are hundreds of possible object classes, and the objects in each class can vary widely. Nevertheless, if an object is sufficiently unambiguous, you have no trouble "instantly" identifying it, even if the slide showing it is displayed for a very short time. Within a fraction of a second after the light from a slide reaches your eyes, you are ready to name the object appearing in the slide.

The neurons in your brain's visual pathway cannot perform long sequences of computations in a fraction of a second; during that time any one neuron can fire at most on the order of 100 times. Evidently, your visual system must be able to perform complex tasks very rapidly through the use of massive parallelism. In particular, it is able to recognize familiar, but unexpected objects, belong to highly variable object classes, in at most a few hundred cycles of its "wetware".

Traditional computational approaches to object recognition fall many orders of magnitude short of this performance. This is not simply because they are implemented on single-processor computers; many of the traditional techniques are inherently sequential, and it is not obvious how they could be speeded up to the required degree even through massively parallel multiprocessing. The purpose of this position paper is to suggest some computational techniques that could serve as a basis for rapid recognition of objects, if implemented on suitable parallel hardware.

This research was supported by the National Science Foundation under Grant DCR-86-03723.

Note: This paper is frankly speculative; many of the ideas in it are still in their formative stages. It is being published in this preliminary form in order to invite comments, criticisms, and counterexamples.

In the real world, object recognition is greatly aided (or occasionally misled) by expectations and by context. Thus a computational model for real-world object recognition should be strongly goal-directed. But as our random slide show experiment demonstrates, rapid, accurate recognition is also possible in the absence of prior expectations or contextual clues. We deal in this paper only with this type of recognition.

Section 2 of this paper reviews traditional paradigms for characterizing and recognizing complex classes of objects, and points out some of their serious limitations. Section 3 presents some conjectures about how humans may characterize such object classes, and Section 4 discusses how objects might be rapidly recognized using appropriate parallel hardware.

2. CLASSICAL PARADIGMS AND THEIR LIMITATIONS

In this section we briefly review classical paradigms for characterizing (or "modeling") classes of objects and for recognizing an object as belonging to a given class, and indicate why these paradigms are inadequate for our purposes.

2.1. Characterizing objects

The traditional approach to characterizing a class of objects is to regard the objects as composed of parts that have given properties and are related in given ways. The parts can in turn be regarded as composed of subparts, etc., but in practice the hierarchy of parts, subparts,... is not very deep. In this paper we will generally ignore the hierarchical nature of object parts. We will also consider primarily *geometric* properties and relations of the parts, but our ideas extend straightforwardly to incorporate properties and relations involving shading, color, or texture.

The parts/properties/relations approach is quite satisfactory for describing manufactured objects composed of well-defined, precisely described parts, but its limitations become apparent if we try to apply it to even the simplest classes of real-world objects. Here, even if the parts are well-defined, it is by no means easy to define the constraints on property and relation values

that characterize a given object class. To give a very simple, two-dimensional example, suppose the objects are alphanumeric characters composed of straight line segments. (Note that this ignores not only shading, color, etc., but also stroke thickness, curvature, gaps, wiggles, serifs, flourishes, etc.) Given a collection of three line segments, what properties and relations must it have in order to be recognized as a block capital "A"? The field of automatic character recognition is over 30 years old, but no one has yet succeeded in completely formulating such a definition.

The difficulty of fully characterizing classes of real-world objects arises from two sources:

a) Such classes generally do not have crisp definitions; they must be defined "fuzzily". In our "A" example, some configurations of three line segments would be recognized by all observers as perfect A's, others would be regarded as imperfect or distorted A's, still others would be rejected by some observers, and so on. (For some additional remarks on the recognizability of alphanumeric characters see Section 5.) Thus it is not always obvious whether a given configuration is or is not an A; we can at best assign it a *degree of membership* in the class of A's.

b) The space of configurations of object parts has many degrees of freedom, even if the parts themselves are simple (here: straight line segments). Thus to characterize the class of A's, we must define its "membership function" over a high-dimensional space. This function does not have a simple form; the property and relation values that give rise to acceptable A's are highly interdependent.

Thus characterizing object classes can be impractical even when the objects are two-dimensional and are composed of well-defined parts (such as straight line segments) for which only a few well-defined properties and relations are relevant. It becomes even less practical when the parts, properties, and relations are not mathematically well defined—e.g., what are the parts that comprise a script capital A, and how are the shapes of these parts characterized?

2.2. Recognizing objects

Suppose we have somehow succeeded in characterizing a class of objects in terms of parts, properties, and relations. Recognizing that an object belonging to the given class appears in an image may still be very difficult. The standard approach to object recognition involves finding appropriate parts in the image (i.e., segmentation); computing properties of and relations among these parts; and looking for a configuration of parts whose property and relation values satisfy the constraints that characterize the given class. It is well known that each of these steps involves major difficulties, some of which will now be discussed.

a) Finding the parts in the image

In an image of a three-dimensional object, the parts of the object may not all be visible, and even if a part is visible, the image shows only a two-dimensional projection of it, seen from a viewpoint that is not known a priori. Thus relating image parts (regions, region boundaries ("edges"), etc.) to object parts is not straightforward.

Even if the object is two-dimensional (e.g., an alphanumeric character), so that its parts are all visible in the image, it may be difficult to find them correctly. Finding regions or region boundaries in a noisy image is difficult, and so is segmentation of a region or boundary into parts based on geometric criteria.

Even if segmentation can be performed correctly, conventional segmentation techniques are generally slow. For example, detecting a long straight line segment by conventional methods requires a number of computational steps on the order of the segment length, which may be hundreds of pixels. Similar remarks apply to segmentation tasks involving more general types of regions, region boundaries, or curves.

b) Measuring properties and relations

For a three-dimensional object, the properties of and relations between projected images of object parts provide only partial information about the three-dimensional properties of and relations between the corresponding object parts, and thus are of limited value in object recognition.

Even if the object is two-dimensional, many of the parts found in the image will be "noise"; for example, an object part may be represented by two or more image regions, or a region may represent a fusion of two or more object parts. Thus the properties of many of the image parts will be meaningless; and the relations between pairs of parts are even less likely to be meaningful. [For example, if half the parts are incorrect, three-quarters of the pairs of parts are likely to be incorrect.]

Even in cases where the properties are meaningful, conventional techniques for computing their values are relatively slow. For example, computing arc length or region area usually requires a number of computational steps on the order of the quantity being measured.

c) Finding configurations that satisfy the constraints

Finding sets of image parts that could represent a given three-dimensional object involves a process of constraint intersection. For each image part, and each object part that could have given rise to it, the position and orientation of the object must satisfy certain geometric constraints. If these constraints have a nonempty intersection for a set of image parts and corresponding object parts, we have evidence that the object is in fact present. This method works best when the geometry of the object is accurately known (e.g., it is a manufactured object), so that the projections of the object onto the image plane can be accurately predicted. It is not clear how well the method works if the object belongs to a highly variable class (e.g., it is an animal or plant of a given type).

Even if the object is two-dimensional, finding it in a noisy image may require extensive search among the image parts, especially if the image parts do not always correspond to object parts due to segmentation errors. The problem can be regarded as one of finding an isomorphic copy of a given (small) labeled graph as a subgraph of a (large) labeled graph (possibly after some merging or splitting of nodes); here the graph nodes are the image parts, the node labels are property values, and the arc labels are relation values. Techniques for finding given subgraphs in noisy graphs have been developed, but they are limited in the amount of noise they can handle.

If we want to be able to recognize objects of many different types in the image, the classical approach requires us to search for them one by one, since we do not know in advance which ones may be present. If the number of object types is large, this sequential search process is too slow. In principle it can be speeded up by hierarchically "indexing" the objects so that, by applying tests of increasing complexity, we can successively eliminate classes of objects; but it is not clear how to do this in practice for a widely diverse collection of highly variable object types.

3. SOME CONJECTURES ABOUT OBJECT DESCRIPTION

We have seen that if the traditional paradigms are used, it is difficult to characterize real-world classes of objects explicitly or to recognize them in an image. Humans, however, can rapidly and reliably recognize such objects. This suggests that humans use simplified methods of characterizing classes of objects, designed so that the constraints defining the classes can be quickly checked when an image is presented.

This section proposes a set of conjectures about the methods that humans may use, when performing rapid object recognition, to describe objects and to characterize classes of objects. As in Section 2, we will emphasize the geometrical level of description, and ignore shading, color, and texture. Many facts about the human visual system will also be ignored, for simplicity; for example, we do not take into account the falloff of resolution away from the center of the field of view, nor the fact that properties are often not computed veridically (illusions).

Conjecture 1

For purposes of rapid recognition, humans represent a three-dimensional object by a set of characteristic views or "aspects", i.e. by a set of commonly occurring two-dimensional projections.

In effect, this reduces the problem of rapid three-dimensional object recognition to a set of two-dimensional problems. The number of aspects needed could in principle be quite large, but ordinarily object orientations are quite constrained. We tend to visualize familiar objects as seen from one of a few standard viewpoints; probably we recognize them rapidly only when seen from approximately those viewpoints. Recognition from other viewpoints may require "mental rotation", which is reported to take on the order of a second or longer.

The constraints on part properties and relations that characterize an object must be quite loose if the object is to be recognized from a viewpoint that is known only approximately, since most property and relation values vary under perspective transformations. In fact, it has been found that the human observer is a "sloppy geometer" [3], and "recognizes" objects even if their representations in an image are not geometrically correct.

The parts themselves must also remain fairly stable under changes in viewpoint. When a change causes major parts to appear or disappear, merge or split, it has given rise to a new aspect. Minor changes give rise to "subaspects"; hierarchical indexing of the aspects would probably be a useful aid to rapid recognition.

Conjecture 2

To a first approximation, humans describe the image of an object (as seen from a given aspect) as consisting of a set of "primitive" parts. There are two types of such parts: pieces of regions and pieces of boundaries.

A primitive piece of boundary is a maximal boundary arc having a simple shape, e.g. straight (i.e., a "side"), convex, concave, etc. (cf. the "codons" of [4]). A primitive piece of region is a maximal subregion having approximate central symmetry (i.e., a blob), or a simple-shaped local symmetry axis and a width function of a simple form (i.e., a ribbon; cf. the "smoothed local symmetries" of [5]). We will not try to precisely define the class of primitive parts here; they are not necessarily the same as codons or smoothed local symmetries.

We conjecture that the primitive parts are the perceptually salient parts of an object, in the sense that object descriptions are formulated in terms of the properties of and relations between these parts. Note that primitive parts are not always unambiguously defined; it may be unclear, for example, whether a nearly straight arc should be regarded as a single boundary piece or as a concatenation of two pieces.

Even in the simplest cases, the parts themselves have parts, e.g. endpoints (for example, a corner is an endpoint of two boundary segments and of a symmetry axis segment), and these subparts too are perceptually salient. Similarly, a blob has a boundary, and a ribbon has "sides" and "ends". In fact, the parts are not truly primitive; they are consistent configurations of subparts.

Parts can be defined at different scales. For example, a wiggly edge is regarded as consisting of many short segments at a fine scale, but may be regarded as a single "straight" edge at a coarser scale. Similarly, a symmetry axis may have many small branches at a fine scale, but may be regarded as a simple arc at a coarser scale; or a

string of small blobs may form a dotted arc at a coarse scale. The parts that play dominant roles in the description of an object are likely to be those that are defined at relatively large scales, comparable to the object size. However, this is not simply a matter of resolution; small parts are perceptually conspicuous if they are isolated.

Conjecture 3
The properties used by humans to describe parts for purposes of rapid recognition are local property values, or simple combinations of such values.

Such properties might include position (of an endpoint or a centroid); arc length (of a boundary or symmetry axis segment); average radius (of a blob) or width (of a ribbon); area; slope (at an endpoint), average slope, or slope of principal axis; average absolute curvature; number of local extrema or zero-crossings of curvature; etc. [We will not discuss here the nature of the coordinate frame with respect to which position and slope are defined.] Maxima or minima rather than sums or averages can also be useful; for example, position extrema define a "box" within which the part is contained. This is not meant to be an exhaustive list, but it indicates the variety of global properties that can be computed by combining local property values in simple ways, e.g. by summation. Other properties can be defined as combinations of these, e.g. the "shape factor" of a blob (area/perimeter^2) or the elongatedness of a ribbon (length/width). Still other properties of parts can be defined in terms of the absence of other, undesired parts; for example, a boundary is smooth if it has no corners.

The topological property of connectedness cannot be defined by combining local property values. On the other hand, it is well known that the connectedness of a region is not perceived immediately unless the region has a sufficiently simple shape. We conjecture that nonconnectedness is perceived immediately only when the parts are clearly separated, e.g. when they are contained in disjoint "boxes".

Conjecture 4
The relations used by humans to describe combinations of parts for purposes of rapid recognition are defined in terms of relative values of properties.

For example, we can specify that two parts are (approximately) equal (with respect to a particular property value), that one is greater than the other, etc. Note that since position is a property, this allows relations of relative position, such as distance, as well as more qualitative relations of direction and distance such as above/below, left/right, near/far, etc. Relative values of relative property values can also be used, e.g. nearer/farther, which involve relative distances, or "between", which involves relative directions.

Some relations between parts cannot easily be expressed in terms of relative property values. Examples are adjacency (of two regions), tangency (of two boundaries), or crossing (of two curves). Note, however, that these relations can usually be associated with the presence or absence of other perceptually salient parts—e.g., if two regions fail to be adjacent along some segment of their boundary, a ribbonlike gap must exist between them, while if two curves cross or touch, angles are created at their intersection point. We conjecture that relations such as adjacency are not perceived immediately, and that nonadjacency is perceived immediately only when it gives rise to a perceptually conspicuous gap.

The topological relation of surroundedness (i.e., the fact that one part is inside another) is also not always perceived immediately. We conjecture that nonsurroundedness is perceived immediately only when there is a clear line of sight (e.g., when some perceptually salient point of the inner object has no point of the outer object to its right), which is a relation of relative position.

Conjecture 5
Humans characterize a class of objects (as seen from a given aspect) using a set of simple unidimensional constraints on individual property and relative property values; they do not use multidimensional constraints.

To give a very simple illustration of this idea, we consider a block capital L, since it is even simpler than an A. A configuration of two straight line segments looks like an L if one is approximately vertical, the other approximately horizontal, and the lower endpoint of the vertical segment approximately coincides with the left endpoint of the horizontal segment. In addition, the ratio of the lengths of the segments should lie in the proper range, with the vertical somewhat longer than the horizontal.

Each of these constraints can be expressed as a *tolerance interval* around an ideal property or relative property value; or, more generally, it can be expressed as a *membership function* from the set of values into [0,1]. For example, the membership function for "approximately horizontal" is defined on the property of slope and has a peak around 0°. (The exact size and shape of this peak need not concern us here.) Similarly, the membership function for the ratio of the segment lengths (vertical:horizontal) has a peak in some range above 1. The membership function for "approximately coincides" can also be defined as a ratio, where the numerator is the distance between the endpoints (i.e., the difference between their positions) and the denominator is the sum (or max) of the segment lengths. [Note that for some properties (position, slope) it is appropriate to define "relative" in terms of differences, while for others (length) it is appropriate to use ratios.]

The constraints on an L listed above are not independent. The slopes of the two segments cannot simply be chosen independently, with one approximately horizontal and the other approximately vertical; in addition,

the slopes must be such that the segments are approximately perpendicular. In a classical pattern recognition framework, this would be expressed by a bivariate membership function (defined on the pair of slopes) of an appropriate form, with the membership value dropping off as the individual slopes depart from their ideal values (0° and 90°) and also as their difference departs from 90°. *We conjecture that humans do not use such bivariate constraints, but rather use sets of redundant univariate constraints.* In our example, in addition to the constraints of approximate horizontality and verticality on the individual segments, we would impose a constraint of approximate perpendicularity, defined by a membership function on the difference of slopes, peaked around 90°. We conjecture that these three constraints are treated by humans as though they were independent, though in fact they are not; and that overall membership is measured by "intersecting" them (i.e., by taking the min of their membership values). Thus in characterizing objects we use a higher number of dimensions than necessary (here: three, even though there are only two slopes); we treat the dimensions as independent even though they are mathematically related, and we treat them all alike even though some of them can be regarded as more basic than others (e.g., the slopes are basic dimensions, while their difference is a "derived" dimension). We do not map the constraints on the redundant dimensions into a subspace having only the minimal necessary number of dimensions, as would be done in the classical approach.

The constraints used in our example have all been of a very simple form, defined by membership functions having single peaks. *We conjecture that humans generally use only such unimodal membership functions.* If a membership function is strongly bimodal, humans regard the object class as consisting of two subclasses, each having a unimodal membership. Functions of complex form are not used; the basic form of a peak is a plateau having simple-shaped dropoffs.

4. AN APPROACH TO OBJECT RECOGNITION

We have conjectured in Section 3 that for purposes of rapid recognition, humans use a simple class of object descriptions, based on typical aspects, perceptually salient parts, and simple property and relative property values, and that they characterize object types by imposing simple univariate constraints on these values. We now suggest how objects characterized in this way can be rapidly recognized using suitable parallel computational techniques.

Our approach consists of three stages:

(1) The image is input to an array of processors having an appropriate type of connectivity; two extensively used examples of such arrays are a hypercube and an exponentially tapering pyramid. This processor array segments perceptually salient parts from the image and measures their property values in time proportional to the logarithm of the image size (the dimension of the hypercube or the height of the pyramid).

(2) The part properties are broadcast to another set of processors that contain object characterizations.

(3) Each of these "object processors" computes the appropriate relative property values and checks whether the constraints defining its object type are satisfied.

Further details about these stages will be given below. Note, meanwhile, that at stage (1), by using hypercube- or pyramid-connected hardware operating in parallel, segmentation and property measurement are performed very rapidly. The broadcasting at stage (2) is basically sequential, but we assume that the properties of only the most conspicuous image parts are broadcast, so that the total amount of data broadcast is limited. Finally, at stage (3) the part descriptions (property values) extracted from the image are checked in parallel against the entire set of object characterizations. [In terms of subgraph isomorphism, here the subgraphs all check the data extracted from the graph, in parallel, looking for copies of themselves; this is the reverse of the traditional procedure in which the graph checks itself for copies of the subgraphs.]

4.1. Part segmentation and property value computation

Segmentation of arbitrary (complex-shaped) regions or boundaries from an image seems to require computation time on the order of the region diameter or boundary length, even if implemented on parallel hardware. If we restrict ourselves to "primitive" (i.e., simple-shaped) boundary arcs or regions, however, such as sides, blobs, or ribbons, divide-and-conquer techniques can be used to segment them from the image in times on the order of the log of the diameter or length, using appropriate parallel hardware. Techniques for performing these types of fast segmentation operations on an image are under intensive study in our laboratory; for a recent review of this work see [6].

We assume that the image, say of size $2^n \times 2^n$, is input to a square array of processors ("cells", for short), one pixel per cell. In a hypercube, each cell is connected to the cells in its row and column of the array at distances 1, 2, 4, . . . , 2^{n-1} (modulo 2^n). In a pyramid, we have n additional arrays of cells of sizes $2^{n-1} \times 2^{n-1}$, $2^{n-2} \times 2^{n-2}$, . . . , 2×2, and 1×1, one above the other, and each cell in a given array is connected to a block of cells in the array below it. The techniques discussed in [6] assume a pyramid of cells, but they have straightforward analogs in the hypercube case.

Our techniques segment perceptually salient parts from the image by constructing trees of pyramid cells in which the root cell of a tree represents the given part and the leaf cells (in the base of the pyramid) correspond to the pixels that belong to the part. Since the pyramid tapers exponentially, the height of a tree (and hence the

time needed to construct it) is proportional to the logarithm of the size of the part. Parts that are large or isolated are represented by roots high up in the pyramid, while small, non-isolated parts correspond to roots lower in the pyramid. Thus our techniques are able to rapidly find perceptually salient parts in the image, and to represent them by roots whose heights in the pyramid are related to their conspicuousness.

We will not discuss here in detail the types of parts that can be segmented from an image using these techniques, but it is suggested in [6] that they include "groupings" of pixels (having given local properties) that satisfy the Gestalt "laws" of similarity, proximity, good continuation, and closure. It is widely agreed among psychologists that in human perception, such parts are generated by "preattentive" processes, and are then used in the process of object recognition. [The recognition process too must be regarded as "preattentive" when we are dealing with unexpected objects and are able to recognize them "at a glance". On the other hand, the grouping processes are probably innate, whereas characterizations of objects must be learned.]

If an image part is represented by a tree of pyramid cells, properties of the part defined by sums, maxima, etc. of local property values can be quickly computed by the cells in the tree, using simple divide-and-conquer techniques, so that the results are available at the root of the tree in time proportional to the tree height, i.e. to the logarithm of the part size.

A cell high up in the pyramid obtains input data from a large block of the image. Such a block may contain or intersect many image parts; many blob-like parts may be contained in it, and many arc-like or ribbon-like parts may pass through it. We assume that the capacity of a cell to represent (pieces of) parts is limited, and that if this capacity is exceeded, the cell preserves only statistical information about the properties of the set of parts that its image block contains. We further assume that the cell tries to group its set of parts into subsets having unimodally distributed property values. In particular, if one of the parts differs sufficiently from all the others, the cell preserves information about the unique part while statistically summarizing the data about the others. Thus unique image parts, even if they are not isolated, are represented by root cells relatively high in the pyramid.

4.2. Broadcasting

We postulate a "readout" process that scans the upper levels of the pyramid and broadcasts the set of property values (or statistical summaries of property values) stored at each cell to the object processors (i.e., the processors that contain characterizations of objects). According to the principles sketched in Section 4.1, the cells on the upper levels of the pyramid contain statistical summaries of the properties of small image parts, as well as properties of individual parts that are large, isolated, or unique. These latter are the parts that are perceptually conspicuous, i.e. whose presence in the image is noticed "immediately".

Our assumption that the readout process starts with the upper levels of the pyramid applies to situations in which there are no prior expectations. Presumably, prior knowledge can be used to control the order of readout, as regards types of parts, sizes, positions, etc.; but goal-directed processing will not be discussed in this paper.

4.3. Constraint checking

When an object processor receives the broadcast data, it computes (for each possible association of the image parts with object parts) the necessary relative property values and checks whether the appropriate constraints are satisfied. Note that relative property values are not computed in the pyramid and are not broadcast; this saves much broadcasting time, and each object processor computes only those relative values that it needs. Relations between parts defined by the presence of other parts (e.g., perceptually conspicuous gaps) are also determined at this stage. (Relations based on the absence of other parts cannot be reliably determined, since the other parts may be present but their properties may not have been broadcast.)

We do not consider here how the object processors are related to one another or how they are interconnected. One can imagine that they are organized in some type of whole/part hierarchical structure in which objects can have subobjects in common, so that the subobjects need only be processed once. On the other hand, a hierarchical organization based on object classes (e.g., "dog" and "horse" are subclasses of "mammal") is less relevant to our object recognition task; the parts segmented from an image may look like parts of a (specific type of) dog or horse, but it is not clear what the parts of a general mammal look like. The organization of our object processors, which are concerned with iconic descriptions of objects, constitutes only one aspect of the organization of semantic memory, which also involves the names of the objects, their class relations and associations, their functional roles, and so on.

Since the image is usually noisy, many of the parts segmented from the image will not correspond to object parts. Thus even if an object is present, its description is not likely to be more than partially satisfied by the image data. We assume that an object is recognized at first glance only if many of its parts are visible in the image, but this is not the case for any other object—i.e., the configuration of parts visible in the image is distinctive. (Recall that we are dealing with images showing only a single object on a blank background.)

In some cases, an object can be recognized based on seeing only a few of its parts, if they are sufficiently distinctive; this capability is exploited by caricaturists. The effectiveness of our approach depends on the fact that our pyramid techniques can rapidly find global image parts that have rich, distinctive descriptions; the parts are not merely local features, which could be rapidly extracted

using conventional techniques.

If enough parts are visible, we recognize an object even though extraneous parts may also be present; for example, we can recognize an object even when someone has scribbled on the image. On the other hand, if the additional parts interfere with the segmentation of the correct parts—for example, if the lines belonging to the object are smoothly extended—the object becomes very hard to detect.

We recognize an object even though its parts may also belong to (or look like) other objects. This seems to cause no problem for rapid recognition; the rivalrous perceptions can be resolved in subsequent analysis.

Once an object is recognized, it becomes possible to reanalyze the image in a goal-directed fashion. This allows us, for example, to modify the segmentation in an attempt to find missing parts (by changing the segmentation criteria or by merging or splitting previously found parts), to resolve inconsistencies, or to examine less perceptually salient parts which represent finer details of the object. Discussion of this goal-directed analysis process is beyond the scope of this paper. [It should be pointed out that goal-directed analysis may play a role even in the perception of unfamiliar objects. If such an object is "recognized" as belonging to a given general class (e.g., the class of polygons, of smooth blobs, etc.), goal-directed methods can then be used to confirm this description, supply missing details, etc.]

5. DISCUSSION AND DISCLAIMERS

The purpose of this paper was to suggest a computational scheme that might account for human performance in recognizing familiar, but unexpected objects "at a glance", i.e. in on the order of 100 neural "cycles". Assuming that an object subtends at least several degrees of visual angle, its retinal image is hundreds of receptor cells across, so that conventional techniques of segmentation and property measurement would require hundreds of computational steps, even if implemented by parallel processing, e.g. on a mesh-connected computer. Our pyramid- (or hypercube-) based approach should reduce this time substantially, since the pyramid height is only about 10, and we can use divide-and-conquer techniques to find the needed image parts and compute their needed properties in tens of (parallel) steps. Only a limited amount of property data need be broadcast to the object processors, which then compute the needed relative property values and carry out constraint checking in parallel.

Our concepts of perceptual saliency for parts, properties, and relations have been based on introspection; we have not considered analysis techniques based on processes that do not seem to be subject to introspection, e.g. processes involving image transforms. We have assumed (at least tacitly) that the parts are extracted in a "pyramid space" derived from the image by resolution reduction, rather than in a transform domain. We have also assumed that the object processors are localized, rather than constituting some type of distributed memory.

We have not intended to imply that there is a clear dichotomy between recognition at a glance and recognition based on deliberate inspection. Some objects will take longer to recognize because they are more easily confused with other objects or because the image is noisier. In terms of our proposed approach, it may sometimes be necessary to obtain more data from the image (from lower pyramid levels, e.g.) before a decision can be made, or it may be necessary to perform some goal-directed reanalysis in an attempt to decide among alternative or rivalrous objects. If the situation is noisy enough or ambiguous enough, prolonged inspection may be necessary. [We have not discussed the noise-resistant properties of our approach, nor how it might be implemented using unreliable components.] Our approach is intended only to demonstrate how recognition might take place very rapidly in a sufficiently clear-cut situation.

At the other extreme, if an object is viewed for a very short period of time (e.g., tens of milliseconds), it is usually not possible to recognize it, but it may be possible to get a general impression of its appearance, e.g. that it is compact or that it is vertically elongated. In terms of our approach, this may result from lack of sufficient time to broadcast the properties of any but the largest parts, so that only crude shape information is available.

Recognition in the absence of prior expectations or context is a very special task; normally one does have expectations, and objects do not occur in isolation. However, this special task is an important one. The visual system is occasionally confronted with surprises, and must handle them rapidly and accurately; it cannot always depend on expectations. It should be recognized, however, that there are many possible levels of expectation; for example, in some situations one might be expecting a certain general class of objects. In fact, some types of tacit "expectations" may always be present as a result of cultural factors.

The absence of expectations, or wrong expectations, can greatly increase the difficulty of recognition. For example, if we expect alphanumeric characters, we can easily interpret odd configurations of lines as characters, and can rapidly distinguish characters from one another; but if we have no prior expectations, or if we are expecting pictures of animals, the configurations may make no sense. As an extreme case of this, many oddly designed typefaces are readable because we expect letters, and can tell (by elimination or from context) which letter is intended; but many of these letters could not be correctly identified in the absence of such context. Similarly, a number of authors have illustrated the complexity of the task of recognizing alphanumeric characters by demonstrating that a very wide variety of patterns can all be recognized as (e.g.) A's; but many of these patterns would in fact not be recognized as A's if we did not know that they were intended to be A's—in other words, they are examples of goal-directed recognition rather than

recognition at a glance.

Humans have a remarkable capacity for recognizing images that they have previously seen. [Hard-to-segment images also become much easier to interpret on subsequent occasions, once they have been successfully segmented.] This does not necessarily involve storing and matching of iconic templates; it may be based on storage and matching of specific sets of relative property values. In any case, rapid recognition of unexpected familiar objects does not depend on having seen the same objects previously; recognition is still immediate even for instances of the objects that we have never seen before.

In conclusion, it should be emphasized that this paper has not presented a report of research results, but only an outline of a proposed research program. Some of the ideas described are under active investigation (e.g., the pyramid approach to segmentation), but others are still at the level of speculations (e.g., the conjectures about object characterization and the parallel object recognition process). It is hoped that the publication of this paper will stimulate exploration of (and improvement on) these ideas both in our laboratory and elsewhere.

REFERENCES

1. T. Pavlidis, *Structural Pattern Recognition*, Springer, Berlin, 1977, Section 6.2.
2. A. Rosenfeld and A.C. Kak, *Digital Picture Processing* (second edition), Academic Press, New York, 1982, Section 12.2.4.
3. D.N. Perkins, Why the human perceiver is a bad machine, in J. Beck et al., eds., *Human and Machine Vision*, Academic Press, New York, 1983, 341-364.
4. W. Richards and D.D. Hoffman, Codon constraints on closed 2D shapes, *Computer Vision, Graphics, Image Processing* **31**, 1985, 265-281.
5. M. Brady and M. Asada, Smoothed local symmetries and their implementation, *International J. Robotics Research* **3** (3), 1984, 36-61.
6. A. Rosenfeld, Some pyramid techniques for image segmentation, Technical Report 1664, Center for Automation Research, University of Maryland, May 1986.

PARALLEL ALGORITHMS FOR COMPUTER VISION ON THE CONNECTION MACHINE

James J. Little, Guy Blelloch, and Todd Cass

Massachusetts Institute of Technology
Artificial Intelligence Laboratory

Abstract

We show how to solve a set of benchmark problems for Computer Vision for the Connection Machine. These problems are intended to comprise a representative sample of fundamental procedures for Image Understanding. The solutions on the Connection Machine embody general methods for filtering images, determining connectivity among image elements, and determining geometry of image elements. We identify a set of low and intermediate vision modules, including edge detection, from zero-crossings of $\nabla^2 * G$ and from the Canny detector, connected component labeling, Hough transforms, and element visibility. More general problems such as graph matching and shortest path calculation are also included. The implementation of these generic modules demonstrates the ability of the Connection Machine to solve vision problems easily and rapidly, using a variety of interesting aspects of the communication structure of the machine.

1 Introduction

The Connection Machine is a powerful fine-grained parallel machine which has already proven very useful for implementation of vision algorithms. Among the existing vision systems on the Connection Machine are binocular stereo [Drumheller86] and optical flow detection[Little86b]. In implementing these systems, several different models of using the Connection Machine have emerged, since the machine provides several different communication modes. The Connection Machine implementation of algorithms can take advantage of the underlying architecture of the machine in novel ways. We describe here several common, elementary operations which recur throughout this discussion of parallel algorithms.

1.1 The Connection Machine

The Connection Machine [Hillis85] is a parallel computing machine having between 16K and 64K processors, operating under a single instruction stream broadcast to all processors (figure 1). It is a Single Instruction Multiple Data (SIMD) machine, because all processors execute the same control stream. Each of the processors is a simple 1-bit processor, currently with 4K bits of memory. There are two modes of communication among the processors: first, the processors are connected by a mesh of wires into a 128×512 grid network (the NEWS network, so-called because of the four cardinal directions), allowing rapid direct communication between neighboring processors, and, second, the *router*, which allows messages to be sent from any processor to any other processor in the machine. The processors in the Connection Machine can be envisioned as being the vertices of a 16-dimensional hypercube (in fact, it is a 12-dimensional hypercube; at each vertex of the hypercube resides a chip containing 16 processors). Figure 2 shows a 4-dimensional hypercube; each processor is connected by 4 wires to other processors. Each processor in the Connection Machine is identified by a unique integer in the range $0 \ldots 65535$, its hypercube address, imposing a linear

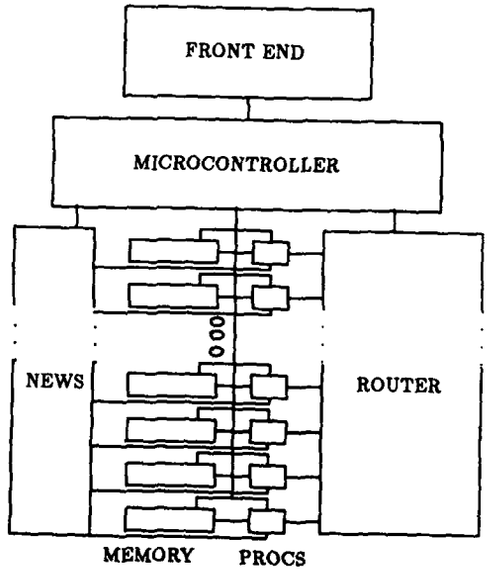


Figure 1: Block Diagram of the Connection Machine

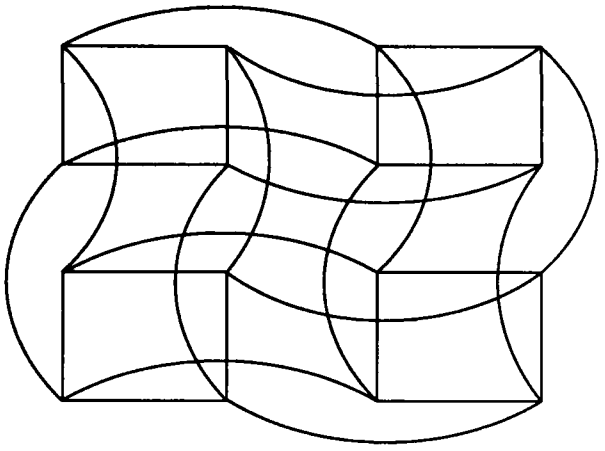


Figure 2: 4-dimensional Hypercube

order on the processors. This address denotes the destination of messages handled by the router. Messages pass along the edges of the hypercube from source processors to destination processors. In addition to local operations in the processors, the Connection Machine has facilities for returning to the host machine the result of various operations on a field in all processors; it can return the global maximum, minimum, sum, logical AND, logical OR of the field.

To allow the machine to manipulate data structures with more than 64K elements, the Connection Machine supports the concept of *virtual processors*. A single physical processor can operate as multiple virtual processors by serializing operations in time, and dividing the memory of each processor accordingly. This is otherwise invisible to the user. The number of virtual processors assigned to a physical processor is denoted by the *virtual processor ratio* (VP ratio), which is always ≥ 1. When the VP ratio is greater than 1, the Connection Machine is necessarily slowed down by that factor, in most operations. The host for the 16K Connection Machine at the MIT AI Lab is a Symbolics 3640 Lisp Machine. In this paper, we assume a 64K Connection Machine. Connection Machine programs utilize Common Lisp syntax, in a language called *Lisp [Lasser86]. Statements in *Lisp programs are compiled and manipulated in the same fashion as Lisp statements, contributing significantly to the ease of programming the Connection Machine.

1.2 Powerful Primitive Operations

Many vision problems must be solved by a combination of several communication modes on the Connection Machine. The design of these algorithms takes advantage of the underlying architecture of the machine in novel ways. There are several common, elementary operations which recur throughout this discussion of parallel algorithms: routing operations, scanning and distance doubling.

1.2.1 Routing

Memory in the Connection Machine is attached to processors, each of which has, at present, 4K bits of local memory. Local memory can be accessed rapidly. Memory of processors nearby in the NEWS network can be accessed by passing it through the processors on the path between the source and destination. At present, NEWS accesses in the machine are made in the same direction for all processors, i.e., to read in one processor, using NEWS access, say, the North, and, in the next, the South, requires two separate accesses. The *router* on the Connection Machine provides parallel reads and writes among processor memory at arbitrary distances and with arbitrary patterns. It uses a packet-switched message routing scheme to direct messages along the hypercube connections to their destinations. This powerful communication mode can be used to reconfigure completely, in one parallel write operation, taking one router cycle, a field of information in the machine. The Connection Machine supplies instructions so that processors can concurrently read and concurrently write a memory address, but since these memory references can cause significant slowdown, we will usually only consider exclusive read, exclusive write (EREW) [Cook80] instructions. We will usually not allow more than one processor to access the memory of another processor at one time. However, in the Hough Transform we will take advantage of combiners since we are guaranteed few and evenly spaced collisions. The Connection Machine can combine messages at a destination, by various operations, such as logical AND, inclusive OR, summation, and maximum or minimum.

1.2.2 Scanning

A powerful set of primitive operations are the *scan* operations [Blelloch86]. These operations can be used to simplify and speed up many algorithms. They directly take advantage of the hypercube connections underlying the router and can be used to distribute values among the processors and to aggregate values using associative operators. Formaly, the *scan* operation takes a binary associative operator

```
processor-number = [0  1  2  3  4  5  6  7]

A                = [5  1  3  4  3  9  2  6]

Plus-Scan(A)     = [0  5  6  9 13 16 18 21]
Max-Scan(A)      = [0  5  5  5  5  5  9  9]
```

Figure 3: Examples of *Plus-Scan* and *Max-Scan*.

```
processor-number = [0  1  2  3  4  5  6  7]

A                = [5  1  3  4  3  9  2  6]
SB (segment bit) = [1  0  1  0  0  0  1  0]

Plus-Scan(A, SB) = [0  5  6  3  7 10 19  2]
Max-Scan(A, SB)  = [0  5  5  3  4  4  9  2]
Min-Scan(A, SB)  = [MX 5  1  3  3  3  3  2]
Copy-Scan(A, SB) = [0  5  5  3  3  3  3  2]
```

Figure 4: Examples of Segmented *Scan* Operations. MX is the Maximum Possible Value.

$\oplus$, with identity 0, and an ordered set $[a_0, a_1, \ldots, a_{n-1}]$, and returns the set $[0, a_0, (a_0 \oplus a_1), \ldots, (a_0 \oplus a_1 \oplus \ldots \oplus a_{n-2})]$. This operation is sometimes refered to as the data independent prefix operation [Kruskal85]. Binary associative operators include minimum, maximum, and plus. Figure 3 shows the results of scans using the operators *maximum* and *plus* on some example data.

The four scans operations *plus-scan*, *max-scan*, *min-scan* and *copy-scan* are all implemented in microcode and take about the same amount of time as a routing cycle. The *copy-scan* operation takes a value at the first processor and distributes it over the other processors. These scans operations can take segment bits that divide the processor ordering into segments. The begining of each segment is marked by a processor whose segment bit is set, and the scan operations will start over again at the begining of each segment. Figure 4 shows some examples.

Versions of the *scan* operations also work using the NEWS addressing scheme - we will call these *grid-scans*. These allow one to sum, take the maximum, copy, or number values along rows or columns of the NEWS grid quickly.

For example, *grid-scans* can be used to quickly find for each pixel in an image the sum of a $2m \times 2m$ square region centered at the pixel. This sum can be determined for all pixels by executing a *plus-scan* along the rows. Each pixel then gets the result of the scan from the processor $m/2$ in front of it and $m/2$ behind it, and by subtracting these two values, each pixel has the sum in its neighborhood along the row. We now executes the same calculation on the columns, giving each element the sum of its square. The whole calculation only requires a few *scans* and routing operations and runs in time independent of the size of m.

We will see several other uses of the *scan* operations later in the paper.

1.2.3 Distance Doubling

Another important primitive operation is *distance doubling* [Wyllie79][Lim86], which can be used to compute the effect of any binary, associative operation, as in *scan*, on processors linked in a list or a ring. For example, using *max*, *doubling* can find the extremum of a field contained in the processors. Using message-passing on the router, *doubling* can propagate the extreme value to all processors in the ring in $O(\log N)$ steps, where N is the number of processors in the ring. Each step involves two *send* operations. Typically, the value to be maximized is chosen to be the cube-address (a unique integer identifier) of the processor. At termination, each processor connected in the ring knows the label of the maximum processor in the ring, hereafter termed the *principal processor*. This serves to label all connected processors uniquely and to nominate a particular processor (the *principal*) as the representative for the entire set of connected processors. At the same time, the distance from the *principal* can be computed in each processor. Figure 5 shows the propagation of values in a ring of eight processors. Each processor initially, at step 0, has an address of the next processor in the ring, and a value which is to be maximized. At the termination of the i^{th} step, a processor knows the addresses of processors $2^i + 1$ away and the maximum of all values within 2^{i-1} processors away. In the example, the maximum value has been propagated to all 8 processors in $\log 8 = 3$ steps.

Step	0	1	2	3	4	5	6	7
0	1	2	3	4	5	6	7	0
	4	1	5	2	11	12	19	3
1	2	3	4	5	6	7	0	1
	4	5	5	11	12	19	19	19
2	4	5	6	7	0	1	2	3
	19	19	12	19	19	19	19	19
3	0	1	2	3	4	5	6	7
	19	19	19	19	19	19	19	19

Figure 5: Distance Doubling, upper entry is the address, the lower is the value

2 Vision Tasks

We catalog a set of vision tasks common to many approaches to early and intermediate vision. Early vision algorithms involve many operations having *spatial parallelism*, where all pixels are operated on in unison, in a spatially homogeneous computation. The classic example is filtering or convolution. A large class of regularization computations can be cast in this framework, when the input data lie on a regular mesh. These spatially parallel computations are easily mapped into operations using only NEWS accesses. When regular, but not spatially homogeneous operation is necessary, operations can be used to produce simple, efficient algorithms. Notably, scans in grid directions can also be used to make more efficient some spatially homogeneous computations such as region summing, described above. Finally, tasks requiring arbitrary, diverse communication among elements can utilize the router's

power. So, we see there are natural classes of operations which map easily into the different communication modes of the Connection Machine. We will first describe edge detection, which in most aspects is a spatially parallel operation, but as we shall see, not all. In these algorithms, an initial mapping of processors to pixels in the image array is assumed.

3 Edge Detection

Edge detection is an important first step in many low-level vision algorithms. It generates a concise, compact description of the structure of the image, suitable for manipulation in higher-level interpretation tasks. There are many edge detectors, in the literature, but all share common aspects of spatial parallelism. Here we will analyze the Connection Machine implementation of Marr-Hildreth edge detection and Canny edge detection schemes.

3.1 Filtering

A fundamental operation in vision processing is filtering the input image. This both removes noise from the image and selects an appropriate spatial scale. Typically, filtering is accomplished by convolution with a filter of bounded spatial extent, often a Gaussian. We have implemented a variety of methods for computing the Gaussian convolution of an image.

An interesting, simple implementation of Gaussian convolution relies on the binomial approximation to the Gaussian distribution. The algorithm described here requires only the operations of integer addition, shifting, and local communication on the 2-D mesh. Since it does not require global broadcasts, large memory, or more than the simple network connections, it could be implemented on a 2-D mesh architecture much simpler than the Connection Machine.

To derive the binomial approximation, consider the discrete signal $x[0] = 1/2$, $x[1] = 1/2$, and $x[n] = 0$ otherwise. The Z transform [Oppenheim75] of this sequence is $(1+z)/2$. Convolution of the two discrete signals $x_1[n]$ and $x_2[n]$ corresponds to multiplication of their respective Z transforms $X_1(z)$ and $X_2(z)$. The result of the convolution $x_1[n] * x_2[n] * ... * x_m[n]$ has Z transform $X(z)^m = (1/2)^m(1+z)^m$ where $x_i[n] = x[n]$ for all i, and $*$ denotes convolution. Thus the sequence resuling from $m-1$ convolutions of $x[n]$ with itself is the inverse Z transform of $(1/2)^m(1+z)^m$. So the resulting discrete signal is just the binomial coefficients to $(1+z)^m$ scaled by $(1/2)^m$. From the Central Limit Theorem, for large m, this sequence approaches a discrete Gaussian.

With each convolution, the peak of the result shifts to the right by one. By advancing the kernel $x[n]$ in space by one, $x'[n] = x[n+1]$, and convolving once with $x[n]$, followed by once with $x'[n]$, the peak of the result remains at $n = 0$. For an even number of such alternations this is equivalent to half as many convolutions with the kernel $y[n] = x'[n] * x[n]$, and so $y[n]$ is given by: $y[-1] = 1/4, y[0] = 1/2, y[1] = 1/4$, $y[n] = 0$ otherwise.

This operations is very simple to implement on a 1 bit processor such as the Connection Machine. Such a convolution amounts to a processor adding 1/2 of its own value to 1/4 of the sum of two of its neighbors, e.g. north and south, or east and west. The scaling can be accomplished simply by right shifting, or possibly reading from the neighboring processors with a one bit offset in the field address. Because a 2-D Gausssian convolution is separable into two 1d convolutions, this can be used to implement an approximation to convolution of a 2-D signal with a 2-D Gaussian filter.

Since the variance of the sequence $y[n]$ is 1/2, the variance of the result of convolving $y[n]$ with itself n times is simply $(m+1)/2$. Thus to approximate a Gaussian distribution with standard deviation σ, $m = 2(\sigma)^2 - 1$ iterations are needed.

This approach is especially suited to the fine-grained architecture of the Connection Machine, where, at present, a multiplication is much more expensive than an addition. Convolution with a Gaussian filter can also be approximated by iterated convolution with a uniform (boxcar) filter of width N and height $1/N$. To approximate a Gaussian with standard deviation σ,

$$\frac{12\sigma^2 - 1}{N^2 - 1}$$

iterations are required. This approximation is useful when σ is large.

The customary implementation of the two-dimensional Gaussian utilizes the fact that the Gaussian is separable, and uses two one-dimensional convolutions. Because this requires multiplications, the cost must be analyzed in terms of them: 16σ multiplies, 8σ additions, and 8σ NEWS accesses. The present implementation of separable convolution should become more efficient than binomial approximation with the simple kernel at $\sigma \geq 10$. Using additional storage, (4σ words), and more clever programming could reduce the time for separable convolution. There are many other tricks in implementing simple filtering operations: for example, boxcar filtering can be implemented using the summing method based on *scanning*, described above, giving dramatic improvements in speed. Further experience will show us how to make these choices.

To implement $\nabla^2 * G(\sigma)$, one often chooses to approximate by the Difference of Gaussians [Marr and Hildreth, 1977], $G(\sigma_1) - G(\sigma_2), \sigma_1 \leq \sigma \leq \sigma_2$. Medioni and Huertas [Huertas85] show how to implement $\nabla^2 * G$ directly as two separable convolutions, which is more efficient than two convolutions with Gaussians as required by the Difference of Gaussians. We directly compute $\nabla^2 * G$ by convolution with a Gaussian followed by convolution with a discrete Laplacian. This is mathematically valid, but is not often

used, presumably since it requires maintaining extra precision in the output of the Gaussian convolution. Because the Connection Machine does not have word or byte boundaries in its local processor memory, it is relatively easy to implement extended precision arithmetic. Therefore it is easy to compute $\nabla^2 * G(\sigma)$ by convolving with $G(\sigma)$, retaining extra precision, and then filtering with the discrete Laplacian:

$$\begin{array}{ccc} 1 & 4 & 1 \\ 4 & -20 & 4 \\ 1 & 4 & 1 \end{array}$$

To detect zero-crossings, each processor need only examine the sign bits of neighboring processors, using NEWS accesses.

3.2 Border Following

Border following begins to use the more flexible communication abilities of the router of the Connection Machine. To analyze this task, we consider two parameters, n, the number of curves in the image, and m, the number of pixels on the longest curve. Each pixel in the Connection Machine can link up with the neighbor pixels in the curve, by examining its 8-neighbors in the grid, in negligible time. Higher vision modules need a unique tag or pointer to refer to a curve as a unit; this is essentially the problem of computing connected components. Further, that label should be known by all the pixels in the curve. So, each pixel on the curve must next be labeled with a unique identifier for the curve. *Doubling* permits the pixels on the curve to select a label, the address of the *principal processor*, for the curve, and to propagate that label throughout the curve in $O(\log m)$ steps.

Then, the total number of curves can be computed by selecting the principal processors, and *enumerating* them using a *scan* operation. The *enumerate* operation returns the number of curves (n).

3.3 Reconfiguring for Output

At this point, the curves have been linked, labeled uniquely, and counted. The structure constructed so far is sufficient to support most operations on curves for image understanding, so we can consider all processing after this to be for output only. This goes against our guideline of keeping all information in the Connection Machine, but is an interesting example of restructuring data in the machine. To output the pixels from the Connection Machine, the points on the curves should be numbered in order to create a stream of connected points. The curve-labelling step, using *doubling*, can be augmented to record the distance from the *principal processor*, as well as its label, during label propagation, at only a slight increase in message length. We can find the length of the longest curve, m, by one global maximum operation. We use sorting to get the points on the curves into a stream order for output. For the i^{th} point in the j^{th} curve, we construct the number $mj + i$ to encode the point's position on the curve and its membership in the j^{th} curve. This yields a unique index for each point. Each point is *ranked* by this value. Points of the j^{th} end up ranked in order of their position on the curve, in the time needed to sort a field of $O(\log m + \log n + 1)$ bits.

The ordered pixels then *send* their (x,y) values to the address given by the rank,in one operation, with no collisions. The (x,y) coordinates of the pixels on the curve will be in sequential order in the processors by cube address. The computation needed for border following depends on $\log m$, the length of the longest curve, and $\log n$, the number of curves in the image; typically, both are proportional to n, in an $n \times n$ image. Labelling and indexing points, and enumerating curves are necessary to construct curves out of individual pixels. Constructing a reconfigured list of the coordinates of edge points is only necessary for output.

4 Canny Edge Detection

The Canny edge detector is often used in image understanding[Canny86]. It is based on directional derivatives, so it has improved localization. Implementing the Canny edge detector on the Connection Machine involves implementing:

1. Gaussian Smoothing $G * I$
2. Directional Derivative $\nabla(G * I)$
3. Non-Maximum Suppression
4. Thresholding with Hysteresis

Gaussian filtering and computing directional derivatives are local operations as described above. Non-maximum suppression selects as edges candidates, those pixels for which the gradient magnitude is maximal in the direction of the gradient. This involves interpolating the gradient magnitude between each of two pairs of adjacent pixels amoung the pixels eight nieghbors, one forward in the gradient direction, one backward. This is accomplished in constant time using the NEWS network.

Thresholding with hysteresis is used to eliminate weak edges due to noise. Two thresholds on gradient magnitudes are computed, *low* and *high*, based on an estimate of the noise in the image intensity. The non-maximum suppression step has selected those pixels for which the gradient magnitude is maximal in the direction of the gradient. In the thresholding step, all selected pixels with gradient magnitude below *low* are eliminated. All pixels with values above *high* are considered as edges. All pixels with values between *low* and *high* are considered edges if they can be connected to a pixel above *high* through a chain of pixels above *low*. All others are eliminated. This is essentially a spreading activation operation; it requires propagating information along curves.

4.1 Histogramming

Estimating the gradient magnitude distribution can be performed by histogramming that value. There are several ways this can be implemented on the Connection Machine

Gradient magnitudes can be quantized for the histogram bucket size, for m buckets. Sorting the gradient magnitudes configures the data thus:

$$\ldots k, k, k, k, k+1, k+1, k+1 \ldots$$

Each processor determines whether it is a lower boundary between elements with value k and the elements with value $k+1$, for all k. A boundary processor sends its cube address to location H_k, in the histogram table, resulting in a cumulative frequency distribution table. To convert this table into a histogram, each H_i subtracts H_{i-1}, or the appropriate lower value when i was not represented in the image. A *copy-scan* can fill in empty elements to simplify this differencing operation.

Histogramming in m buckets by *counting* involves stepping from $0 \leq i \leq m$, selecting processors where intensity $= i$, and counting the number of selected processors (H_i), using global counting operations. This needs m counting operations. When m is less than 64, this can be more efficient than histogramming by sorting.

Finally, when only one value in the distribution is needed, say, finding the k^{th} percentile, estimation via probing can be used. Here, a binary search on a set of m values is performed in $O(\log m)$ global counting operations, until the value i, $0 \leq i \leq m$ is found for which k percent of the values are $\leq i$. For $m = 256$, this requires only 8 counting operations. A more reliable estimate of the distribution involves computing the entire histogram, but probing may suffice for many applications.

Computing a Hough Transform, which will be discussed later, is a form of histogramming. There, we will be able to take advantage of *a priori* information on the distribution of values to devise a fast algorithm. When the form of the distribution of image values is known, similar methods can be applied for general histogramming.

4.2 Propagation

In thresholding with hysteresis, the existence of a *high* value on a connected curve must be propagated along the curve, to enable any *low* pixels to become *edge* pixels. This requires a number of operations depending on the length of the longest segment above *low* which will be connected to a *high* pixel. Only pixels above *low*, which survive non-maximum suppression, are considered. Each pixel can, in constant time, find the one or two neighboring pixels with which it forms a connected line, by examining the state of all 8 neighbors in the NEWS network. All pixels above *high* are marked as *edge* pixels. In the current implementation, the program iterates, in each step marking as *edge* pixels any *low* pixels adjacent to *edge* pixels. This uses NEWS connections. When no pixel changes state, the iteration terminates, taking a number of steps proportional to the length m of the longest chain of *low* pixels which eventually become *edge* pixels. Using *doubling*, propagating a bit to indicate the *edge* property, changes this dependence to $O(\log m)$. For most practical examples, propagating in the NEWS network is faster than using the asymptotically optimal *doubling* procedure.

5 Connected Component Labeling

A fast practical alogrithm for labeling connected components in 2-D image arrays using the Connection Machine has been developed by Lim et al. [Lim86]. The algorithm has a time complexity of $O(\log N)$ where N is the number of pixels. The central idea in the algorithm is that propagating the largest or smallest number stored in a linked list of processors to all processors in the list takes $O(\log L)$ time, where L is the length of the list, using *doubling*.

In the algorithm (see [Lim86] for more details), the label of a connected (4-connected) component is the largest processor address (i.e. processor id) of the processors in the set. The complexity of the algorithm is measured in terms of N, the length of the longest boundary in the image. The whole algorithm takes $O(\log N)$ routing cycles on the Connection Machine.

We have devised a connected component algorithm utilizing *scan* operations along grid-lines. In each phase of his algorithm, the label of a region, as specified by the processor with maximum cube-address, is propagated left, right, up and down, with a *max-scan* operation. The number of phases of this algorithm depends on the complexity and the alignment of figures in the image. Its worst-case behavior originates from an image containing long ellipsoidal regions, oriented along diagonals.

6 Hough Transform

The Hough transform is frequently used in image analysis to determine the existence of straight lines in an image [Ballard and Brown,1981]. The Generalized Hough Transform similarly is used to determine the parameters of the position and orientation of a known object from an image. The Hough Transform is an archetypical algorithm in that it demonstrates the accumulation of global information about the image from individual elements in the image. The Hough transform computes an accumulator array of values, where the (i, j) entry records the number of pixels lying on a line with parameters (i, j).

Let us parametrize lines by the angle of the normal vector, i, and the perpendicular distance from the origin, j. The Hough Transform table will be stored in a matrix of processors, indexed by (i, j). Computing the Hough Trans-

form involves i separate operations, each of which computes the Hough Transform for a particular angle, $\theta(i)$. For each angle, broadcast $cos\theta(i)$ and $sin\theta(i)$ to each of the processors. Each processor then computes the scalar product of its (x, y) address in the grid with the normal vector described by the broadcast pair:

$$j = x\cos\theta(i) + y\sin\theta(i)$$

Each processor then knows an (i, j) for itself. We can count votes by *sending* a 1 from each active pixel, summing at the destination processor. If we were simply to *send* each vote to the appropriate destination in the table, there would be many collisions, messages arriving at one processor, especially when (i, j) does in fact represent a line. This can be very slow when the number of collisions is large (≥ 64). To avoid this, we can use a clever trick, suggested by Mike Drumheller (personal communication), to reconfigure the processors.

In the reconfiguration method, each pixel computes its location on a linearization of the processors in the machine by lines normal to the specified angle (see figure 6). Bold lines mark the normal lines. The pixels on the first normal line are *sent* to positions $0 \ldots P0$, those on the second line go to $P0+1 \ldots P1$, and so on up to $N-1$. Each pixel then has a unique address, sequential along the normal lines, in the machine. Each pixel can *send* its value to the processor with its number, in one router cycle (there are no collisions). The pixels then lie, in linear order in the machine, according to their position on the normal lines. A *boundary processor* is one which occurs at the beginning of one of the normal lines; it sets a *segment-bit*. Then a special *plus-scan* operation, using segments, can accumulate the numbers for the histogram in the boundary processors. One *send* operation can collect the values into the histogram. This suffices to construct a column $(i, j), 0 \leq j \leq max$, of the histogram. Each angle requires some computation to

1) compute the scalar product

2) compute an address along scan lines

One *send*, followed by a *scan*, followed by a *send* completes the process for a column. The procedure describe here uses unique addresses for the linearization step. The routing hardware incurs little penalty for having up to 32 collisions per destination. A randomizing strategy is thus feasible, provided the number of collision is minimized, and easier to program. Each pixel computes a random value at the start. For each i, a pixel computes a location based on j and the random value, using a linearization along normal lines similar to the deterministic procedure. Locations for all pixels with the same j are confined to a contiguous block of processors in the Connection Machine. The sizes of the blocks can be calculated to allow enough space to reduce the average number of collisions close to zero. The votes, when they arrive, are summed, using a built-in capability of the router. Then the votes are tallied as before using a special *plus-scan* operation, with segments. This will achieve the same result, probably with less overhead for computing addresses.

If the Hough Transform were derived from oriented edge fragments, rather than pixels, only one iteration would be needed. Each processor can generate the (i, j) value from its (x, y) location and the edge orientation θ. Each processor then *sends* its vote, with the *:add* combiner to (i, j). The maximum number of processors in a 256x256 image voting for the same (i, j) would be no more than $256\sqrt{2}$, which would take no more than one step of the previous Hough Transform algorithm.

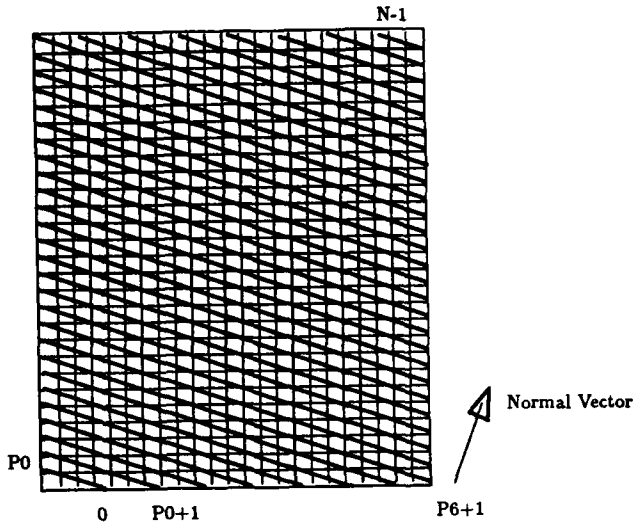


Figure 6: Mapping for the Hough Transform

7 Geometrical constructions

A variety of geometrical algorithms were designed for the Connection Machine, including several convex hull algorithms and a Voronoi Diagram algorithm.

7.1 Convex Hull

There are four planar convex-hull algorithms that have been suggested for the Connection Machine. As with serial convex-hull algorithms, which one is the most practical will depend on the specifics of the problem. We will describe each one briefly. All of these algorithms for m points require $O(n)$ processors.

The first algorithm is based on a Concurrent Read Exclusive Write (CREW) PRAM algorithm described in [Atallah85, Agrawal85]. Since the Connection Machine does not perform well on concurrent reads, we have shown elsewhere that their algorithm can be modified to work with exclusive reads and *scans* [Blelloch87]. This algorithm for n points has the optimal asymptotic time bound of $O(\lg n)$. Although this algorithm has the best asymptotic bound

of the algorithms we mention, it has a large constant and therefore might not be the most practical algorithm.

The second is a parallel version of the QUICKHULL method [Preparata85]. This algorithm can be implemented using a small constant number of router cycles and scans per step [Blelloch87]. As with the serial QUICKHULL, for m hull points, this algorithm runs in $O(\lg m)$ steps for well distributed hull points, and has a worst case running time of $O(m)$.

The third is a parallel version of Graham's sequential algorithm [Preparata85] and requires $O(\log^2 n)$ routing cycles for n points [Little86a]. The final algorithm is a parallel version of the Jarvis march algorithm [Preparata85], requiring only a few arithmetic operations, plus a global maximum computation per hull point [Little86a]. Although this algorithm requires $O(m)$ steps for m hull points, it might be the most practical for some applications since the constant is small and the number of points on the hull is often small.

7.2 Voronoi Diagrams

Aggarwal et al. [Aggarwal85] describe a $O(\log^3 N)$ algorithm for computing the Voronoi diagram of N points in parallel using the CREW (Concurrent Read Exclusive Write) model. For $N = 1000$, this is 1000 steps, each of which needs at least one routing cycle. Since the Connection Machine has the NEWS network, a set of mesh connections among the processors, a brush-fire method can be easily implemented. One can argue that in many vision applications the coordinates of the points are restricted to the range of the resolution of the camera coordinate system, in which case working at image resolution is acceptable.

In the proposed algorithm, each pixel in the mesh is labeled with the (i, j) location of the point closest to its (x, y) coordinate, which we may take to be its center. The result of this labeling is accurate only at the grid-spacing; each grid intersection is labeled with its closest point. For the L_1 and L_∞ metrics, propagation is simple. By using 4-connected or 8-connected neighbors, we can make the arrival time of a propagated label be identical to its L_1 or L_∞ distance. The Euclidean-distance metric is more problematic, and we must propagate the (x, y) of a point as well as its index. The Voronoi region around a point can be labelled in D steps, where D is the diameter of the largest Voronoi region.

The Delaunay triangulation is the dual of the graph of the Voronoi diagram. We say a pixel is and *edge pixel* if any neighbors do not have the same label. To generate the adjacency information for the vertices in the Delaunay triangulation, label each edge pixel by the indices of the points whose regions it bounds; for edge connecting i to j, generate, for $i \leq j$, $i * N + j$. This generates a number of $\log^2 N$ bits, for N points. This number is a compressed representation for the edge. Sort these numbers, and then eliminate all those whose lower neighbor (by hypercube address) is the same. This eliminates multiple entries for each edge. Then, each edge in the Voronoi Diagram has a unique processor identified with it; this can rapidly be transformed into a useful graph representation on the Connection Machine, in which the edges adjacent to a vertex are stored in a contiguous block of processors, and edges (i, j) contain the address of the processor dedicated to (j, i). Each edge is thus stored twice.

Propagation is the most costly operation in this algorithm, and depends of the diamater of the largest Voronoi region. Trial examples with 1000 randomly distributed points had average maximum diameter approximately 12. The additional computation to label edges and generate the graph representation will take less than one propagation cycle. The result of this implementation is only roughly correct; certain thin Voronoi regions can cause errors. However, these may not be of any practical consequence. Charles Leiserson and Cynthia Phillips (personal communication) have developed a simple extension to this algorithm which corrects these faults.

8 Triangle Visibility

Certain systems in vision need to determine visiblity of scene elements. It is interesting to consider the performance of the Connection Machine on such a problem, even though the problem is essentially in the domain of computer graphics. For example, let the input be a set of m opaque triangles in three-dimensional space, selected at random with each coordinate in some range. Find the vertices of the triangles that are visible from the origin.

The problem can be parallelized over the set of triangles or the set of vertices. Even though there are three times as many vertices, the description of a triangle is larger, and the increase in communication costs overcomes the reduced number of elements. A triangle *shadows* all vertices which lie in the triangular cone formed by the origin and the edges of the triangle, and which are behind the plane containing the triangle. The volume in space defined by this criterion is described by 4 linear inequalities, from the bounding half-spaces. Each triangle, in a preprocessing step, generates the four plane equations. A vertex can then be tested for visibility by evaluating these equations for its (x, y) coordinates.

The following formulation uses multiple copies of the triangles. The problem can be parallelized by copying the triangles k times in the memory (64K) of the Connection Machine, where

$$k = 65536/m$$

This divides the machine into k subsets of processors. Each triangle processor will handle up $n = (ceiling(3m^2/65536))$ points. Triangles 0 through k occupy processors 0 through k (cube address), and so forth. The descriptions of the triangles must be generated; this needs several vector sub-

tractions and cross-products to compute normal equations for planes. The computed triangle descriptions comprise 4 plane equations,

$$A_i x + B_i y + C_i z + D = 0$$

each of which contains 4 32-bit numbers; the entire description is 512 bits long. The descriptions of all m triangles can be *copy-scanned* to replicate them k times, and then *sent*, in one step, to the correct processors. Then, points are *sent* to the sets of triangles against which they are to be tested. The first n points are sent to processors $0 \ldots n-1$, the next n to processors $m \ldots m+n-1$, and so forth.

Segments bits are inserted at the termination of each set of triangles, In each testing step, the description of the point at the beginning of each set of points is *copy-scanned* across the set of triangles. All triangles test the active points in parallel. Then, the descriptions of the points are *sent* left. This brings a new point to the beginning of each section of triangles, ready to be copied to all the triangles in the next step. Since there are $n = (ceiling(3m^2/65536))$ steps, the total time required depends on n.

Many of the triangles in this example are themselves completely occluded by larger triangles. We suggest a first filtering step so that small triangles which are occluded by large triangles can be eliminated before the problem is partitioned into groups. Each triangle is projected onto a projection plane, anywhere in the visible region, orthogonal to a line of sight from the origin. Then the projection plane is subdivided into a $k \times k$ grid. Each triangle can determine all squares it covers completely. At the intersection points in this grid, each triangle determines its z-value. Each triangle should cover, on average, one-sixth of the rectangles. Now, each triangle generates a copy of itself for each rectangle it covers; choose k so that

$$k^2/6 * m \leq 65535$$

where m is the number of triangles, so that each triangle-rectangle pair can be allocated to a processor. Using $m = 1000$, k is 18. The triangle-rectangle pairs contain the *zmin*, *zmax* of each triangle in that rectangle. Organize the processors contiguously by rectangle. Then *scan* the triangles to find the minimum of the *zmax* covering the rectangle. Eliminate all triangles in a rectangle whose *zmin* is greater than this *zmax*; these triangles cannot be seen. This should significantly reduce both the number of triangles, and, consequently, the number of vertices. Then, using the process described above, the number of iterations should be significantly smaller.

9 Graph matching

The problem is to compute the list of the occurrences of (an isomorphic image of) a small graph H as a subgraph of a larger graph G. We will outline a method to distribute the matching process among the processors of the CM. A similar solution for objection recognition is described in [Harris86]. We utilize dynamic allocation of processors to matchings. A partial matching is contained in a processor. At each step in the graph matching algorithm a matching (processor) acquires the information necessary to determine all legal successors. It then finds processors to continue with the new matchings, and is then returned to the pool of free processors. The information concerning the graphs can be stored in several ways in the memory of the Connection Machine. We can store the adjacency list for a vertex in G as a bit vector in a processor. The graph G can be stored, with many copies, throughout the CM. Each matching processor can access these copies randomly, so that contention among the processors is minimized. The description of the smaller graph H can be stored in each matching processor, as well as the partial matching it is expanding.

Initially no processors are allocated. We use *rendezvous allocation*[Hillis85] to assign processors to matchings. The order in which vertices in H are matched to G can be precomputed to maximize the number of vertices in H adjacent to the next vertex to be expanded. In each phase of matching generation, a matching at level k, $1 \leq k \leq |H|$, must expand itself to all legal successor matchings at the next level. Matching processors may be expanding at many different levels, since resource limitations may delay expansion until some processor fails, and is returned to the pool. To expand itself, a matching must know, first, the neighbors of H_{k+1}, and, second, the vertices in G to which those neighbors have been matched. These data allow the matching at level k to prune its expansion, generating only legal successors. The description of H is stored locally in each processor. To recover the neighbors of H_{k+1}, each processor steps through the description of H, until they encounter the $k+1^{th}$ entry, and then records the contents of this entry. This step finds the neighbors of the new vertex in H.

Each expanding matching examines the neighbors of H_{k+1} to determine the nodes in G to which they have been matched. The neighbors of each such vertex in G must be retrieved from the distributed representations of G, using a *send* operation. Now, we must compute the intersection of these bit vectors, describing all possible nodes in G adjacent to the matches in G of neighbors of H_{k+1}. This can be done in time linear in the number of nodes in G, but such bit operations are fast. Then we exclude from the intersection all nodes already matched in the current matching, leaving the possible expansions in G. All are legal, that is, the nodes in G to be matched are unmatched, and are adjacent to existing constraining matches from H. If this set is empty, the matching fails.

The entire computation to expand a matching has several parts, each taking, at most, time linear in the size of G. Then, each matching requests processors for its successors from the free pool, in a processor allocation step. The description of a partial matching at the k^{th} level can be

distributed to the successor nodes during allocation. The matching processor then returns itself to the free pool of processors. A priority mechanism can be implemented to favor matchings which are nearer completion. The overhead of allocation and distribution should be no more costly than the entire computation to determine legal successors.

The total throughput of this algorithm can be measured in terms of the number of partial matchings generated in each step. The critical factor in this problem is to control the number of active matchings so as to allow expansion, without create blocking. The process should monitor itself to record the average number of successors at each level, allowing good control of allocation. This discussion by no means solves this difficult problem; there are many thorny issues of control and task allocation yet to be analyzed completely.

10 Minimum-cost path

The problem is, given two vertices P,Q of G, an undirected graph with nonnegative real-valued weights on the edges, to find a path from P to Q along which the sum of the weights is minimum.

The graph can be represented as an adjacency list in the CM. The algorithm, a CM implementation of Dijkstra's algorithm, is given in [Hillis85]. Each step in computing the shortest path consists in each vertex sending to each of its neighbors the distance from the source to itself plus the length of the connecting edge along which the message is sent. Each step involves a *send* operation, using the router. The receiver compares all incoming values and selects the minimum.

Consider sending messages only when the distance from the source is less than infinity (some initial value for all processors). This reduces the number of conflicts at many stages. Initial experiments require only a few router operations per step. The number of steps depends on the diameter (the length of the longest path in the graph explored). The algorithm stops when no processor changes its value as the result of the messages it has received.

11 Summary

We have identified several powerful communication patterns which allow us to utilize the Connection Machine to design efficient algorithms for vision. The powerful communication methods permit construction of extended image structures in logarithmic time. Schemes can easily arrange data in the Connection Machine so that rapid distribution of information using the scan mechanism allows efficient algorithms. These algorithms represent a subset of those currently designed for or implemented on the Connection Machine. Other general algorithms for vision are described in [Harris86] and [Little86a].

We have described the algorithms in terms of fundamental operations of the Connection Machine. We will give a quick outline of the comparative running times of the primitives we have been discussing. Let us take as a time unit the time for a 1-bit operation. The actual time required by this operation will change as the Connection Machine evolves. In these units, then, accessing the memory of a neighboring processor via the NEWS network takes 3 units per bit. Global operations, such logical or and logical and need less than 50 units. Counting all active processors requires 200 units. Routing and scanning uses 500 units. Adding 16-bit quantities takes 20 units, while multiplying them takes 250 units. The relative speed of these operations may change in future versions of the Connection Machine. In another document[Little86a], we give more detailed descriptions of these algorithms and their running times, on the present machine. In actual time, for example, our present implementation of Canny edge detection, operating on a 256x256 image, from image to linked edge pixels, takes less than $50ms$. Many of the other algorithms described above achieve similar times, and show speedup of several orders of magnitude over present implementations on serial computers.

12 Acknowledgments

Support for the A.I. Laboratory's artificial intelligence research is provided in part by the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research contract N00014–85–K–0214.

References

[Aggarwal85] A. Aggarwal, B. Chazelle, L. Guibas, C. O'Dunlaing and C. Yap, "Parallel Computational Geometry", *Proc. 25th IEEE Symp. Found. of Comp. Sci.*, 1985, 468-477.

[Blelloch86] G. Blelloch, "Parallel Prefix vs. Concurrent Memory Access", Technical report (in preparation). Thinking Machines Corporation, Cambridge, Massachusetts, 1986.

[Blelloch87] G. Blelloch and J. Little, "Convex Hull Stuff", submitted to Third Annual Symposium on Computational Geometry, 1987.

[Canny86] J.F. Canny, "A Computational Approach to Edge Detection", IEEE Transactions on Pattern Analysis and Machine Intelligence, Nov. 1986, Vol. 8, No. 6, 679-698.

[Cook80] S.A. Cook, "Towards a Complexity Theory of Synchronous Parallel Computation", Int. Symp. ueber Logic und Algorithmic zu Ehren vo Proffessor Ernst Specker, Zurich, 1980.

[Drumheller86] M. Drumheller and T. Poggio, "Parallel Stereo", Proc. of IEEE Conference on Robotics and Automation, San Francisco, 1986.

[Harris86] J.G. Harris and A.M. Flynn, "Object Recognition Using the Connection Machine's Router", *Proc. IEEE 1986 Conf. Computer Vision and Pattern Recognition*, 1986, 134-139.

[Hillis85] D. Hillis, *The Connection Machine*, MIT Press, Cambridge, 1985.

[Lasser86] C. Lasser, "The Complete *Lisp Manual", Thinking Machines Corporation, Cambridge, Massachusetts, 1986.

[Little86a] J.J. Little, "Parallel Algorithms for Computer Vison", AIM-928, MIT AI Laboratory, November 1986.

[Little86b]J.J. Little and H. Bulthoff "Parallel Computation of Optical Flow", AIM-929, MIT AI Laboratory, November 1986.

[Lim86] W. Lim, A. Agrawal and L. Nekludova, "A Fast Parallel Algorithm for Labeling Connected Components in Image Arrays", submitted to the International Conference on Computer Vision, 1987.

[Huertas85] A. Huertas, and G. Medioni, "Detection of Intensity Changes with Subpixel Accuracy using Laplacian-of-Gaussian Masks", to appear in IEEE Transactions on Pattern Analysis and Machine Intelligence.

[Oppenheim75] A.V. Oppenheim and R.W. Schafer, *Digital Signal Processing*, Prentice-Hall, Englewood Cliffs, New Jersey, 1975.

[Overmars81] M.H. Overmars and J. Van Leeuwen, "Maintenance of Configurations in the Plane", *Journal of Computer and System Sciences*, **23**, 166-204.

[Preparata85] F.P. Preparata and M.I. Shamos, *Computational Geometry - An Introduction*, Springer, New York, 1985.

[Wyllie79] J.C. Wyllie, *The Complexity of Parallel Computations*, TR 79-387, Department of Computer Science, Cornell University, Ithaca, New York, August 1979.

SOLVING THE DEPTH INTERPOLATION PROBLEM

ON A FINE GRAINED, MESH- AND TREE-CONNECTED SIMD MACHINE

Dong J. Choi

Department of Computer Science
Columbia University
New York, N.Y. 10027

Abstract

This paper discusses solving the depth interpolation problem on a particular parallel architecture (a fine grained, mesh- and tree-connected, SIMD machine). Many low level computer vision problems, including depth interpolation, can be cast as solving a symmetric positive definite (SPD) matrix. Usually, the resulting SPD matrix is sparse. In a previous result, we showed how the adaptive Chebyshev acceleration method for sparse SPD matrices could be run on this particular architecture. Here, we show how the conjugate gradient method can be run under same framework. Lastly, we show simulation results for fairly reasonably large synthetic images from two methods, and compare them with the results from one of the commonly used basic iterative methods, the Gauss-Seidel method. We also detail our future plans.[1]

1. Depth Interpolation

Human perception is a vivid one of dense and coherent surfaces in depth. This suggests that there exists a visible-surface reconstruction process that transforms sparse information into a dense surface representation. With the sparse depth and/or orientation constraints generated from low level visual processes, a depth interpolation process would compute the depth of the visible surfaces at every point explicitly.

Grimson formulated one approach to the depth interpolation problem [Grim 81]. He suggested an "interpolation" method. Given a set of scattered depth constraints, the surface which best fits the known constraints is that which passes through the known points exactly and minimizes the expression referred to as the quadratic variation of the surface. He used a gradient descent method to find such a surface and slow convergence rates were observed in his work.

Terzopoulos worked further on surface representation [Terz 84]. Instead of Grimson's "interpolation" approach, he proposed an "approximation" method where the discrete potential energy functional associated with the surface is minimized. In his formulation, known depth and/or orientation constraints contribute as spring potential energy terms.

The discrete form of the visible-surface reconstruction problem is described as the solution of a large sparse linear system of equations. The nonzero coefficients of each equation is specified as summations of computational molecules. Given the depth constraints and the orientation constraints, a set of computational molecules computes the nonzero coefficients of the linear system by local computations. Because of the symmetric nature of the computational molecules, it can be easily shown that the resulting matrix is symmetric. Furthermore, Terzopoulos shows the stronger result that the matrix generated is symmetric and positive definite (SPD). The matrix is also sparse. Even for nodes which are sufficiently distant from a boundary where the depth is discontinuous, they interact with only 12 neighbors, all of them at most only 2 nodes away. Terzopoulos used a multi-grid approach with the Gauss-Seidel relaxation method at each relaxation sweep to speed up the convergence rate.

We follow the Terzopoulos' formulation on visible surface reconstruction and use the computational molecules proposed by him. However, we present an alternative depth interpolation process using the theoretically better iteration methods, which speed convergence and are amenable to certain classes of parallel computers. In our previous paper [Choi 85], we used the adaptive Chebyshev acceleration method. In this paper, we present another efficient method, the conjugate gradient method.

2. Conjugate Gradient Method

The depth interpolation problem has been cast as solving a large system of linear equations[2]

$$Ax = b \tag{1}$$

where $A = A^* > 0$ is an $n \times n$ hermitian[3] and positive definite matrix [Wozn 80].

We solve the equation (1) iteratively by constructing a sequence $\{x^{(i)}\}$ converging to the solution $\alpha = A^{-1}b$. Let $B = B^* > 0$ be a matrix which commutes with A: $BA = AB$.

Let $\|x\|_B = \|B^{1/2}x\| = (Bx, x)^{1/2}$, where $\|x\| = (x, x)^{1/2}$.

Let $x^{(0)}$ be an initial approximation and consider a class of iteration methods for which the error formula satisfies the relation

$$x^{(i)} - \alpha = W_i(A)(x^{(0)} - \alpha),$$

where W_i is a polynomial of degree at most i and $W_i(0) = 1$. We seek the polynomials W_i such that the error $e_i = \|x^{(i)} - \alpha\|_B$ is minimized. The solution is given by the orthogonal polynomials defined as follows. Let

$$x^{(0)} - \alpha = \sum_{j=1}^{m} c_j \xi^{(j)}$$

[1]This research was sponsored by the Defense Advanced Research Projects Agency under contract N00039-84-C-0165.

[2]In this paper, we use x to denote the depth vector. In a previous paper, we used u.

[3]Since the matrix A is both real and symmetric, it is hermitian.

where $\xi^{(j)}$ is an eigenvector of A associated with the eigenvalue λ_j: $A\xi^{(j)} = \lambda_j\xi^{(j)}$, $\|\xi^{(j)}\| = 1$, $0 < \lambda_1 < \lambda_2 < \cdots < \lambda_m$, with $m \le n$ and $c_j \neq 0$ for $j = 1, 2, \ldots, m$. From the orthogonality of the polynomials W_i it follows that they satisfy a three-term recurrence formula. This form is defined as follows:

$$W_0(\lambda) = 1,$$

$$W_1(\lambda) = 1 - c_0\lambda,$$

$$W_{i+1}(\lambda) = W_i(\lambda) - c_i\lambda W_i(\lambda) - u_i\{W_{i-1}(\lambda) - W_i(\lambda) + c_i\lambda W_i(\lambda)\}, \quad i \ge 1,$$

where

$$c_i = \frac{(W_i, W_i)}{(\lambda W_i, W_i)},$$

$$u_0 = 0,$$

$$u_i = \frac{(W_i - c_i\lambda W_i, \frac{1}{\lambda}(W_{i-1} - W_i) + c_iW_i)}{(W_{i-1} - W_i + c_i\lambda W_i, \frac{1}{\lambda}(W_{i-1} - W_i) + c_iW_i)}$$

From this we get the three-term recurrence formula for the sequence $\{x^{(i)}\}$,

$$r^{(i)} = Ax^{(i)} - b,$$

$$z^{(i)} = x^{(i)} - c_ir^{(i)},$$

$$y^{(i)} = x^{(i-1)} - z^{(i)},$$

$$x^{(i+1)} = z^{(i)} - u_iy^{(i)}, \tag{2}$$

where

$$c_i = \frac{(r^{(i)}, B(x^{(i)} - \alpha))}{(r^{(i)}, Br^{(i)})},$$

$$u_0 = 0,$$

$$u_i = \frac{(y^{(i)}, B(z^{(i)} - \alpha))}{(y^{(i)}, By^{(i)})}, \quad i \ge 1. \tag{3}$$

In exact arithmetic, the conjugate gradient method (2), (3) converges in m steps.

For $B = A$ we minimize $\|A^{1/2}(x^{(i)} - \alpha)\|$. This corresponds to the *classical conjugate gradient* method. After further simplification, we have

$$c_i = \frac{(r^{(i)}, r^{(i)})}{(r^{(i)}, Ar^{(i)})},$$

$$u_0 = 0,$$

$$u_i = \frac{(y^{(i)}, Az^{(i)} - b)}{(y^{(i)}, Ay^{(i)})}, \quad i \ge 1. \tag{4}$$

On examination of the equations (2) and (4), we find that we seem to need four matrix multiplications. But two matrix multiplications, $Ay^{(i)}$ and $Az^{(i)}$, can be eliminated with substitutions. Now we have

$$c_i = \frac{(r^{(i)}, r^{(i)})}{(r^{(i)}, Ar^{(i)})},$$

$$u_0 = 0,$$

$$u_i = \frac{(y^{(i)}, r^{(i)} - c_iAr^{(i)})}{(y^{(i)}, Ax^{(i-1)} - Ax^{(i)} + c_iAr^{(i)})}, \quad i \ge 1. \tag{5}$$

We show how this method is easy to implement in the following architecture, and investigate its efficiency.

3. Architectural Background

As we have remarked in our previous paper, a parallel architecture support to meet the particular structure of our application demands following characteristics:

- fine grained
- SIMD
- local communication
- global communication.

In this section, we derive a model of SIMD computation that has all necessary characteristics. Various features of this model will be extracted separately from several actual SIMD machines that have been built.

The time complexity analysis of parallel machines involves two factors: the internal computational speed of each processing element (PE) and the communication speed to move around data between the PEs.

The computational speed of each PE depends on the complexity of the hardware circuitry built into it. In computer vision applications, we have a tremendous amount of data to be processed. Therefore, it forces the designer to design each PE as simple as possible to accommodate more and more PEs. Nevertheless, we need good computational capability as well, such as the floating point calculations as required in this application.

One such PE design which made effort to meet these computational requirements is that of the Massively Parallel Processor (MPP). For the 128 × 128 square mesh, the actual execution speed of 470 million addition operations per second, 291 million multiplication operations per second, and 165 million division operations per second have been reported with 32-bit floating point data format in [Gilm 83, p. 166]. We took these numbers and converted them into equivalent machine cycles in Table 3-1. We took into account of 100 nanosecond machine cycle time of the MPP.

In MPP, NON-VON and Connection Machine, mesh communication instructions, which handle local communications, execute in single machine cycle. We assumed a single bit data path between PEs and a floating point data is moved from the memory of one PE to the other PE's memory. For the transfer of each bit, we assumed a 5 machine cycle sequence: read a bit, send it through mesh connection, and then write it. Therefore, for 32-bit floating point data, it will take 160 machine cycles for completion.

For global communications, we assumed the tree topology of NON-VON. We assumed that the instructions which carry out the tree communications between adjacent levels execute in 2 machine cycles following the experience of NON-VON chip and prototype system design/implementation efforts. We assumed again a single bit data path between PEs and a floating point data is moved from the memory of one PE to the other PE's memory. For the transfer of each bit, we assumed a 6 machine cycle sequence: read a bit, send it through tree connection, and then write it. Therefore, for 32-bit floating point data, it will take 192 machine cycles for completion.

In the Connection Machine, global communication is handled by Boolean n-cube topology. For the prototype with the size of 256×256 mesh, the communications bandwidth of 5.0×10^{10} bits/second for the FFT Pattern has been reported in [Hill 86, p. 72]. There is a single bit data path between chips and it will take 63 machine cycles to move a single bit to the cube neighbor. We took into account of 4 MHz clock and 4,096 routers. The reason why we have such a big number is that the messages are delivered across each dimension in sequence for every *petit cycle* in the Connection Machine. For a 32-bit floating point data, it will take 2,016 machine cycles.

Our discussion is summarized in Table 3-1. This model of SIMD computation will be used throughout this paper.

4. Parallelization

4.1. Storage Space Analysis

In the adaptive Chebyshev method, we need 16 1-bit flags and 16 floating point numbers for the storage space of each PE. Assuming 32 bits for the floating point number representation, we need 528 bits per PE. In the conjugate gradient method, we need 16 1-bit flags and 21 floating point numbers. Here, we need 688 bits per PE.

4.2. Time Complexity Analysis

Suppose that we have $s \times s$ square mesh at the leaf of the tree.

For the adaptive Chebyshev acceleration method, we show the number of operations in *local* and *global* computations and then totals for each operation in Table 4-2. By *local* computations we mean those which require internal computations carried out inside of PE and optional mesh communications with nearby neighbors. The computation of two vectors, δ and $u^{(i+1)}$, falls into this group. The matrix computation, $Gu^{(i)}$, is an embedded step of the computation of the vector δ. By *global* computations we mean those which require tree communications for global summary and other internal computations carried out prior to or during global summary. The computation of three vector norms are put in this group.

Under the column designated "TOTAL", the number of operations are multipled by the number of machine cycles which were defined before in Table 3-1. In summary, the total number of machine cycles required for each iteration of the adaptive Chebyshev acceleration method is given by $4392 \times (\log_2 s) + 16453$. For the tree with 128×128 square mesh at the leaf, the total number of machine cycles is 47197.

We show the result for the conjugate gradient method in Table 4-4. The computation of four vectors and two matrix computations fall into *local* computations. The matrix computations, $Ax^{(i)}$ and $Ar^{(i)}$, are embedded steps of the computation of the vector $r^{(i)}$ and the coefficient c_i, respectively. The computation of four inner products to compute two coefficients, c_i and u_i, are put in *global* computations.

The total number of machine cycles required for each iteration of the conjugate gradient method is given by $5856 \times (\log_2 s) + 32307$. For the tree with 128×128 square mesh at the leaf, the total number of machine cycles is 73299.

5. Simulation Results

We have rewritten the NON-VON simulator to handle the floating point operations. It has been rewritten in FORTRAN 77 and transported to IBM 4381. It now handles up to a 128×128 square image.

In our simulation work, the synthetic image was a constant depth plane ($\alpha = 1.0$). The shape of the boundary of the plane was a square, with size 128×128. The depth constraints were scattered randomly over the plane. The density of the depth constraints were 15%, 30%, and 50%.

The root mean square error (RMSE) at the ith iteration is defined as follows:

$$RMSE^{(i)} = \left(\left(\sum_{j=1}^{n} [x_j^{(i)} - \alpha_j]^2\right) / n\right)^{1/2}.$$

In Table 5-1, we show the number of iterations i to attain the specified RMSE. The results are tabulated side by side for three different iteration methods, the conjugate gradient, the adaptive Chebyshev acceleration, and the Gauss-Seidel method. We observe that the conjugate gradient method performs best in the sense that it takes the least number of iterations. The adaptive Chebyshev acceleration method comes next and the Gauss-Seidel performs worst. But we should note that each step of the iteration of the first two methods is completely parallelized so that overall execution is much faster compared to the Gauss-Seidel method where the computation is done in serial fashion. As the depth constraints become sparser, the depth interpolation problem itself becomes inherently harder to solve and takes more iterations. Even here, the degradation in the Gauss-Seidel method turns out to be the worst.

We have derived before that the total number of machine cycles per iterations are 47197 for the adaptive Chebyshev acceleration method and 73299 for the conjugate gradient method, respectively. In Table 5-2, we show the normalized number of iterations for two methods. For the adaptive Chebyshev acceleration method, we use the same numbers as in Table 5-1. For the conjugate gradient method, we have multiplied the number of iterations in Table 5-1 by 73299 / 47197 = 1.5530. After normalization, the conjugate gradient method performs still better than the adaptive Chebyshev acceleration method. This is in part due to the errors in the initial estimates of the eigenvalues. In the first few iterations, the conjugate gradient method performs much better, but as more computations are done the estimate of the eigenvalues (in this case, the largest eigenvalue only) gets better. Thus, the difference of the number of iterations between the conjugate gradient method and the adaptive Chebyshev acceleration method gets smaller.

We have summarized other numerical values in Table 5-3. There, the measures are listed for three different densities of depth constraints for each iteration method. They are the number of iterations; the minimum, the maximum, and the average of the final depth values; and the 2-norms of the error vector. Furthermore, we have listed $\|A^{1/2}(x^{(i)} - \alpha)\|$ for the conjugate gradient method and the final estimate of the largest eigenvalue, M_E, for the adaptive Chebyshev acceleration method.

When we examine the final depth values, they show another aspect of the depth interpolation problem becoming harder as the depth constraints become sparser. With comparable or sometimes better average values, the minimum and the maximum values deviate further from the solution. This phenomenon is common to all three iteration methods. For example, for the conjugate gradient method, we have smaller values of $\|A^{1/2}(x^{(i)} - \alpha)\|$, the quantity being minimized in this method, as the depth constraints become sparser. Nevertheless, we have comparable 2-norms of the error vector and the average values, and worse minimum and maximum values. In the adaptive Chebyshev acceleration method, the initial estimates of the smallest and the largest eigenvalues were $-\|G\|_\infty$ and 0.0, respectively.

6. Conclusion and Future Plans

In this paper, we showed how the conjugate gradient method can be applied to SIMD machines for those computer vision problems where the resulting matrix is SPD.

We have analyzed the space and time complexity of two iteration methods based on our SIMD model derived from actual machines built. In particular, we analyzed the computational and the communication costs of parallel computing. Also, we have analyzed two modes of communications, local and global, necessary for local interactions and global summary, respectively.

In this paper, a synthetic image with constant depth was described. Other synthetic images such as sphere or cylinder will be pursued in future. Also, we plan to run our algorithms to real images such as range data from scanner.

Acknowledgements

The author is grateful to J. R. Kender, G. W. Wasilkowski, and D. E. Shaw for their advice and support during this work.

References

[Choi 85] Choi, D. J. and Kender, J. R.
Solving the Depth Interpolation Problem with the Adaptive Chebyshev Acceleration Method on a Parallel Computer.
In *Image Understanding Workshop*, pages 219-223. 1985.

[Gilm 83] Gilmore, P. A.
The Massively Parallel Processor (MPP) : A Large Scale SIMD Processor.
In *Proceedings of SPIE*, pages 166-174. 1983.

[Grim 81] Grimson, W. E. L.
From Images to Surfaces.
MIT Press, 1981.

[Hill 86] Hillis, W. D.
The Connection Machine.
MIT Press, 1986.

[Lee 85] Lee, D.
Contributions to Information-based Complexity, Image Understanding and Logic Circuit Design.
PhD thesis, Columbia University, October, 1985.

[Shaw 84] Shaw, D. E.
Organization and Operation of a Massively Parallel Machine.
Technical Report, Columbia University, October, 1984.

[Terz 84] Terzopoulos, D.
Multiresolution Computation of Visible-surface Representations.
PhD thesis, Massachusetts Institute of Technology, January, 1984.

[Wozn 80] Wozniakowski, H.
Roundoff-Error Analysis of a New Class of Conjugate-Gradient Algorithms.
Linear Algebra and its Applications, 29:507-529, 1980.

[Youn 81] Young, D. M. and Hageman, L. A.
Applied Iterative Methods.
Academic Press, 1981.

Operations	Data Type	Execution speed (machine cycles)
Addition, Subtraction	32-bit fl. point	348
Multiplication	32-bit fl. point	563
Division	32-bit fl. point	993
Mesh Communication	1 bit	1
Mesh Communication	32-bit fl. point	160
Tree Communication	1 bit	2
Tree Communication	32-bit fl. point	192

Table 3-1: Speed of Typical Operations

Operations	*local*	*global*	TOTAL (machine cycles)
Addition, Subtraction	16	$6(\log s)$	$(6(\log s)+16)\times 348$
Multiplication	11	1	12×563
Division	1	0	1×993
Mesh Communication	16	0	16×160
Tree Communication	0	$12(\log s)+3$	$(12(\log s)+3)\times 192$

Table 4-1: Summary of Operations (Chebyshev Accel. Method)

Operations	*local*	*global*	TOTAL (machine cycles)
Addition, Subtraction	30	$8(\log s)+3$	$(8(\log s)+33)\times 348$
Multiplication	18	5	23×563
Division	2	0	2×993
Mesh Communication	32	0	32×160
Tree Communication	0	$16(\log s)+4$	$(16(\log s)+4)\times 192$

Table 4-2: Summary of Operations (Conjugate Gradient Method)

RMSE	Conjugate Grad.			Chebyshev Accel.			Gauss-Seidel		
	50%	30%	15%	50%	30%	15%	50%	30%	15%
0.5	5	8	15	27	35	54	29	51	111
0.2	12	20	36	41	59	98	72	130	295
0.1	20	31	54	55	79	128	106	195	459
0.05	27	42	73	65	93	157	141	264	647
0.02	37	57	103	79	120	206	190	363	949
0.01	45	69	129	94	136	250	228	444	1230
0.005	52	81	152	104	160	296	268	530	1554
0.002	62	98	178	118	182	353	322	653	2023
0.001	70	109	199	135	207	383	364	751	2393

Table 5-1: Root Mean Square Error

RMSE	Conjugate Gradient			Chebyshev Accel.		
	50%	30%	15%	50%	30%	15%
0.5	7.8	12.4	23.3	27	35	54
0.2	18.6	31.1	55.9	41	59	98
0.1	31.1	48.1	83.9	55	79	128
0.05	41.9	65.2	113.4	65	93	157
0.02	57.5	88.5	160.0	79	120	206
0.01	69.9	107.2	200.3	94	136	250
0.005	80.8	125.8	236.1	104	160	296
0.002	96.3	152.2	276.4	118	182	353
0.001	108.7	169.3	309.1	135	207	383

Table 5-2: Root Mean Square Error (normalized)

Results from the Conjugate Gradient Method

d	i	$x^{(i)}_{min}$	$x^{(i)}_{max}$	$x^{(i)}_{avg}$	$\|x^{(i)}-\alpha\|$	$\|A^{1/2}(x^{(i)}-\alpha)\|$
50%	70	.984603	1.002782	.999994	.126194	.0143727
30%	109	.980571	1.003009	.999988	.130898	.0096369
15%	199	.979287	1.007600	.999997	.125836	.0053095

Results from the Adaptive Chebyshev Acceleration Method

d	i	$x^{(i)}_{min}$	$x^{(i)}_{max}$	$x^{(i)}_{avg}$	$\|x^{(i)}-\alpha\|$	M_E
50%	135	.985364	1.000801	.999687	.124797	.991694
30%	207	.979400	1.000410	.999826	.126349	.996296
15%	383	.971733	1.001309	.999929	.127170	.998905

Results from the Gauss-Seidel Method

d	i	$x^{(i)}_{min}$	$x^{(i)}_{max}$	$x^{(i)}_{avg}$	$\|x^{(i)}-\alpha\|$
50%	364	.988639	1.000352	.999474	.127688
30%	751	.981034	1.000412	.999687	.128378
15%	2393	.971534	1.001434	.999900	.127979

Table 5-3: Other Results

SURVEY OF PARALLEL COMPUTERS*

Hong Seh Lim and Thomas O. Binford

Artificial Intelligence Laboratory, Computer Science Department,
Stanford University, Stanford, CA 94305, U.S.A.

Abstract

Computer vision systems process large volumes of data and require huge computing power. This renders conventional sequential computers I/O bound or compute bound. Computers with parallel processing capability are designed to address these problems. We survey some of these parallel computers, both general purpose and those oriented towards computer vision. Some of the surveyed computers are in production and can be purchased from the manufacturers; others are in various stages of development in research laboratories. Rather than describing each computer completely, this survey emphasizes special features associated with each computer. Tables of comparisons of the surveyed computers are given.

1 Introduction

Parallel processing has been successfully used to reduce the time of computation for a wide variety of applications. The processing of large amounts of data, the need for real time computation, the use of computationally expensive operations, and other demands that would make a task too time-consuming to perform on conventional sequential computers have motivated computer architects to consider parallel computer design. To help readers understand different kinds of parallel computers, we explore various parallel computers available by examining special features of each computer.

This paper is divided into three main parts. The first part discusses some general considerations involved in the design of parallel processing, including architectures and interconnection networks. The second part outlines the main features of the surveyed computers. They are arranged in alphabetical order by the name of the computer. An appendix, compares the surveyed computers by number of processors, processing power, memory size, architecture, availability, chip used, and software environment.

The DARPA image understanding architecture workshop (1986) provided detail information on the performance of some of the following systems on IU "benchmark" problems.

2 General Considerations

2.1 Architectures

Most computer architectures for parallel processing can be classified as either single instruction multiple data (SIMD) structures or multiple instructions multiple data (MIMD) structures. A few computers have multiple SIMD units that can operate independently, and they are classified as multiple SIMD (MSIMD) structures.

A SIMD computer typically consists of N processing units, an interconnection network, and a control unit. Each processing unit has its own processor and memory. The interconnection network provides interprocessor communication. The control unit broadcasts instructions to the processors. Each active processor executes the same instruction simultaneously, using data taken from its own memory.

A MIMD computer consists of a set of independent processing units and an interconnection network. Each processing unit has an independent instruction stream and operates on independent data. The interconnection network provides communication between processors.

2.2 Interconnection Networks

The interconnection network is crucial to running parallel computers efficiently. We discuss some of the most commonly used networks in parallel computers. They include cross-bar, pipelined, bus, K-dimensional hypercube, Omega, and grid networks.

A cross-bar network allows any processor to access data from all other processors. Thus, there is never

*This work was supported by the Defense Advanced Research Projects Agency under contract number N00039-84-C-0211

contention for communications resources. This scheme requires N^2 switches to connect N processors. However, it becomes too costly to implement when the number of processors is large.

The pipeline or one-dimensional systolic array network is a simple one-dimensional nearest-neighbor connection. The function of each processing unit is specified by the host according to a specific problem or class of problems. Once set up, each processing unit performs the same operation on every datum sequentially. The cost of this network is proportional to the number of processors.

A bus network uses a common shared-data channel for communication. With a bus, any processor may transmit data to other processors through the common data channel. However, since only one processor can transmit at a time, bus contention becomes the major problem of this network. Cache and local memories decrease load on a bus and extend utility to processors. Bus-oriented architectures are prominent in commercial systems.

A K-dimensional hypercube network connects $N = 2^K$ processors. Processors can be viewed as corners of a cube in K-dimensional space with connections as edges of the cube. This network has a high data bandwidth, which grows as NlogN, and a low message latency whose worst case is logN. This network is adaptable to other topologies, such as ring, tree, and two- and three-dimensional meshes.

An N x N Omega network consists of logN identical stages. Between adjacent stages is a perfect-shuffle interconnection. Each stage has N/2 switch boxes under independent box control. The bandwidth is linear in N and the latency is logarithmic in N.

A two-dimensional nearest-neighbor network allows communication between nearest neighbors arranged in a grid. It is particularly suitable for SIMD architecture. However, communication between distant processors is expensive (proportional to the square root of the number of processors). The complexity of the network increases proportionally according to the number of processors. This is a common architecture, augmented in some machines by other interconnection for distant processors.

3 Surveyed Computers

3.1 Balance

The Balance 21000 parallel computer system is a MIMD system offered by Sequent Computer System Inc.. It provides from four to thirty tightly-coupled 32-bit microprocessors with performance which ranges from 2.8 to 21 MIPS. Up to 256 users can be supported simultaneously.

National NS32032 microprocessors are used. Each is supported by a floating-point unit, a memory management unit, and a 8-Kbyte cache. The cache, together with its interrupt control logic, minimizes bus contention among processors. This results in linear speedups on the system as additional processors are added. The whole system can deliver up to 2.25 MFLOPS.

All processors share a common memory pool and a single copy of DUNIX, a parallel version of the UNIX operating system. The memory is interleaved and contains up to 48 Mbytes. It is equally accessible by all processors and peripherals.

A proprietary 32-bit system bus connects all processors and memories. It can sustain an effective data transfer rate of up to 26.7 Mbyte/second. The I/O subsystems can be Ethernet or Multibus interfaces.

The DUNIX operating system dynamically balances the load of the system and distributes multiuser workloads evenly across processors for maximum system throughput. It allows processes to access mutually exclusive hardware directly.

Sequent has a software library that supports parallel programming in C, FORTRAN, Pascal, and Ada.

3.2 Butterfly

The Butterfly Parallel Processing Computer is a product of Bolt Beranek and Newman Inc.. The basic configuration of this MIMD computer starts with four processors; it can be expanded in single processor increments to 256 processors which yield up to 250 MIPS. It has a butterfly switch for interconnecting all processors described below.

Each processor and its memory are located on a single board called a Processor Node (PN), which is the basic computing element of the Butterfly computer. Each PN uses a standard Motorola MC68020 microprocessor, capable of executing one million instructions per second. The microprocessor is augmented by the MC68881 floating point computational unit. Each PN also contains a microcoded control processor that provides interprocessor communication, synchronization, and support for parallel processing.

Each node may have memory size from one Mbyte to four Mbytes, with a maximum system-wide shared memory of one Gbyte. It can independently execute its own sequence of instructions, referencing data as specified by the instructions. Though memory is local to the PNs, each processor can access remotely any memory in the system using the Butterfly switch.

The Butterfly switch implements interprocessor com-

munication using techniques similar to packet switching. Its topology is similar to that of the Fast Fourier Transform Butterfly and it implements a subset of the hypercube network. The switch has a latency of logN and the bandwidth through each processor-to-processor path in the switch is 32 Mbits/second.

The Butterfly I/O system is distributed among the PNs. Any node can have an I/O board that supports data transfer at a maximum rate of two Mbytes/second.

Butterfly application software development is done on a VAX or Sun work station running Berkeley 4.2 UNIX. The system supports C and FORTRAN 77. A multiprocessing Common Lisp system, that will be compatible with the Symbolic 3600 series of Lisp work stations, is under development.

3.3 CAPP

The Content Addressable Parallel Processor (CAPP) is a SIMD computer built at the University of Massachusetts. Its goal is to explore the applications of content addressability and parallelism. The computer consists of two main parts—the central control unit and the parallel processor.

The central control unit is responsible for broadcasting instructions to the parallel processor and for loading and unloading data in the processor. It also serves as an interface between the host and secondary data storage devices. The central control unit is pipelined, and instructions can be prefetched. It contains a ROM with commonly needed micro-coded instructions and a small program memory for storing user programs.

The parallel processor contains 262,144 cells arranged as a 512 x 512 array. Each cell consists of 32 bits of static memory, an ALU, and four one-bit static tags. One of the tags controls whether the cell is active.

Cells are connected to their nearest four neighbors. Because the pin count on the chip is limited, an 8:1 multiplexing scheme was used for communication between chips. Cells on the edge are processed in three ways, first as dead edges connecting no neighbor, second wrapped around, or third wrapping around but offset by one cell, to make a linear array.

3.4 CLIP

Cellular Logic array for Image Processing, CLIP, is a family of array processors developed at University College, London. CLIP7 is the most recent system constructed in the CLIP family. It has improved image resolution to 512 x 512 pixels as compared to 96 x 96 pixels in CLIP4. CLIP7 is implemented as a scanned array of 512 x 4 processors. Each processor deals with data from 128 pixels during scanning.

The original CLIP4 chip has a bit-serial circuit that is flexible but lacks individual power. The CLIP7 is based on a custom-designed integrated circuit, which includes a 16-bit ALU running at 100ns/cycle. This multibit circuit is more appropriate than the bit-serial circuit for image processing on grey scale data.

Each processor has 128 x 256 bits of storage, organized as 4,096 bytes. A local register allows the construction of a system in which each processor in the array has a degree of local autonomy. Externally applied signals control normal chip operation. However, the content of a local register can affect operation if conditional operation is selected.

The CLIP7 chip has a single processor. All data lines are available externally, in particular the neighborhood interconnection lines. This means it is possible to build assemblies of processors having various connection networks.

Data buses in the chip are 16 bits wide, whereas external data accesses are through 8-bit ports. Neighborhood interconnections between processors use single lines because of packaging requirements. Serial transmission of propagation data imposes a 10

All user interactions with the CLIP7 system are through the minicomputer host, which runs the UNIX operating system. The host supports IPC and C programming languages. A control pipeline receives information from the host. Pipelining permits overlapped fetching and execution of instructions.

CLIP7 is provided with a direct video data input channel via TV camera, A/D converter, and a frame store. It is connected to a Winchester disc for data storage. Its effective bandwidth is between 20 and 40 Mbytes/second, allowing storing or retrieving an image in 100ms.

3.5 Connection Machine

Thinking Machines Corp. builds the Connection Machine computer, which has 65,536 processors providing a raw computing power of 1000 MIPS. The computer can be configured to a much larger number of logical processors by creating a two-dimensional array of virtual processors on each physical processor.

The computer can be divided into four equal subparts. Each works at a separate front end, and each has a separate instruction stream executing in a quarter of the system's processors. Thus it becomes a MSIMD computer.

Each processor is a bit-serial processor that operates on two data values specified by each instruction. It has four Kbits of memory (32 Mbytes for the computer). The memory is divided into a data area and a stack

area, and it is bit addressable. Sixteen physical processors are grouped on a chip.

Two ways of communication between processors are available. First, each processor is wired to its neighbors to the North, East, West, and South by the NEWS network. Second, a "Boolean n-cube" network, the Connection Machine Router, connects each of the 65,536 physical processors to 16 other physical processors whose binary addresses are different in just one of the 16 bits. The Router allows a full message to be sent from any processor to any other. The sender processor simply needs to know the address of the destination processor.

Data are exchanged between memory and the front end in three ways. "Read-slice" reads a single bit from the memory of each of a series of consecutive processors. "Read-processor" moves a single field between the front end and a single processor. " Read-array" moves fields between the front end and a set of contiguous processors.

A microcontroller expands macro-instructions from the front end into nano-instructions, which are broadcast to all virtual processors. Processors have the option of "sitting out" some instructions depending on the one-bit Context Flag in each processor.

The Connection Machine provides an assembly-level language REL-2. It also supports parallel versions of C and Lisp.

3.6 FAIM-1

FAIM-1 is a multiprocessing system developed to support concurrent symbolic program development and execution. It consists of many autonomous processing elements, called Hectogons, arranged in a hexagonal mesh. A 19-processor prototype computer is under development at Schlumberger Palo Alto Research.

Each Hectogon has six subsystems connected by the System Bus (SBUS). The Evaluation Processor (EP) is a stack-based processor responsible for evaluating computer instructions. The Switching Processor (SP), which is responsible for context switching, sets up the next runnable task, while the EP evaluates the currently active task. The Instruction Stream Memory (ISM) stores, decodes, and handles instructions to the EP. It runs concurrently with the EP and effectively provides a two-stage pipeline mechanism. The Scratch Random Access Memory (SRAM) is a four-ported local data memory, which provides concurrent access to the EP, SP, SBUS and Post Office. A parallel associative memory for matching structures is the Pattern Addressable Memory (PAM) which stores and matches S-expression structures of symbols and words. Physical delivery of inter-Hectogon messages is carried out through the Post Office subsystem. It delivers messages concurrently with program execution in the processor, detects congestion dynamically, and makes its routing decisions accordingly.

Each Hectogon communicates by sending messages. Fault-tolerant message routing is made possible by the multiplicity of paths over which a message may be routed to its destination.

The FAIM-1's communication topology consists of two levels. The lower level has a number of Hectogons interconnected to form processing surfaces. These surfaces in turn can be interconnected to form a multi-surface configuration on the higher level.

Within each surface in the lower level, the Hectogons are arranged in a regular hexagonal mesh. When wire leave a processing surface at the periphery, they are folded back onto the surface using a three-axis variant of a twisted torus. This wrapping scheme provides a minimal switching diameter for a hexagonal mesh. For a 19-processor surface the worst-case communication requires at most two hops.

Multiple surfaces can also be interconnected using a hexagonal mesh. Multi-surface configurations have the advantage of increased on-surface locality. The communication diameter of the entire system is decreased when compared to a single-surface instance of a similar number of processors. A 58,381-processor computer, arranged on a hexagon of side 140, has a diameter of 139. On the other hand, a 58,807-processor computer, arranged on a 9-surface configuration, each surface forming a hexagon of side 10, has a diameter of 89. Thus, using a multi-surface configuration, the communication diameter is reduced by 50.

3.7 FX/series

Alliant Computer System Corp. offers the FX/series parallel processing system. It can be expanded in the field to provide 94 MFLOPS peak performance.

The architecture of the FX-8 system includes two main resource classes—Interactive Processors (IPs) and the computational complex.

The IP runs interactive user jobs and the operating system in parallel. It maintains system responsiveness and frees the computational complex to concentrate on compute-intensive applications. Each IP is a Multibus containing a virtual memory address translation unit, an I/O map, and both a console and a remote diagnostic serial port.

The IP interfaces with the IP cache, which provides access to global memory. Each IP has 512 Kbytes of local memory and accesses global memory through a

virtual memory architecture. Virtual address space for users is two Gbytes. Up to 12 IPs can be configured as interactive load increases.

The computational complex consists of up to eight processors, called Computational Elements (CEs). Each processor can work independently and delivers 11.8 MFLOPS peak performance. Because the hardware schedules and synchronizes multiple CEs, the performance speedup delivered to a single application approaches the number of CEs installed.

The CP cache serves as an interleaved high-speed physical memory buffer for the computational complex. Two cache modules with four cache ports provide a four-way interleaved 128-Kbyte cache with a maximum bandwidth of 376 Mbytes/second. A crossbar interconnect dynamically connects up to eight computational elements with up to four cache ports.

Physical memory in Alliant systems uses 256 Kbytes of dynamic RAM and is expandable in 8-Mbyte modules up to a maximum of 64 Mbytes.

The Concentrix operating system implements the Berkeley 4.2 UNIX operating system. It supports a multiuser environment and parallel processing without programmer intervention. The system manages two types of jobs and dynamically schedules them on available processors as long as work remains. Compute-intensive jobs take priority on the computational complex. Interactive user jobs, I/O, and other operating system activities are scheduled on any available IP or otherwise idle computational complex.

Concentrix supports FX/FORTRAN, Pascal, C, and Alliant assembler. The FX/FORTRAN fully implements the FORTRAN 77 programming language. C and Pascal languages are supported only in a single computational element.

The FX/FORTRAN compiler identifies those sections of code that can be executed concurrently by multiple CEs during compilation. It optimizes a loop as much as possible, but suppresses optimization wherever the optimized code might produce results that differ from the unoptimized code. Thus, it requires no source code reprogramming from the user. The compiler inserts special concurrency control instructions into the program stream to identify these sections. This concurrency is self-scheduling and controlled by hardware at execution time. Additional CEs can be added without recompiling or relinking of programs. The hardware concurrency control allows a program to initiate, synchronize, and suspend concurrent processing with minimum overhead.

3.8 GAPP

Geometric Array Parallel Processor, GAPP, was originally developed by the Martin-Marietta Corp. and marketed commercially by NCR Corp.'s Microelectronics Division. It is a SIMD processor with 6 x 12 processing elements (PEs).

Each of the PEs contains an 1-bit ALU, 128 bits of RAM, and four registers. It operates with a 10-MHz clock and takes 25 clock cycles to add two 8-bit numbers. With all the PEs running, the processor can deliver 921 million additions per second.

72 PEs are arranged in a 6 x 12 array in the current version of the chip. The chip is fabricated with a 3-um double-layer metal CMOS process, and it is currently housed in a ceramic 84-lead pin-grid array.

Connecting each PE to its neighbors on the north, south, east, and west are bidirectional communication lines. In addition, a separate I/O communication bus allows data to be input from the south end of the array and output to the north without interfering with computations within the ALU.

The implementation of a control store lets the processor receive a set of instructions from the host and store them, freeing the host for other tasks. The control store operates in conjunction with a sequencer, which watches for and maintains the correct sequence as the processor performs its instructions.

The processor is programmed with a sequence of instructions that, when compiled by an assembler, directs the appropriate control signals to every cell in the array. Up to five commands (four for each of the four registers and one for the RAM) can be executed simultaneously on every instruction cycle. A software library of macro-cells forms the basis of a high-level command set for the processor.

3.9 iPSC

Intel Scientific Computers offers iPSC as a multiprocessor system that can operate concurrently with as many as 128 independent processing units connected as a hypercube network.

The iPSC system consists of two major functional elements—the cube and the cube manager.

Each node of the cube is a board-level microcomputer with high-speed versions of the Intel 80286/80287 microcomputer chip sets. It has its own memory of 512 Kbytes, expandable to 4.5 Mbytes. Each node also contains eight bidirectional communication channels managed by dedicated communication co-processors. Seven of these channels are physically linked to other nodes and serve as dedicated channels. The eighth channel is a global Ethernet channel that

provides direct access to and from the Cube manager for program loading, data I/O, and diagnostics. It has a 10-Mbit/second channel for internode serial communication.

The cube manager provides a high-level interface. It serves as the local host for the cube and provides the communication/control software and the system diagnostic facility. It runs the XENIX operating system, a version of UNIX, with Lisp, FORTRAN, C, and assembly language.

The iPSC-VX is a vector concurrent system built upon the basic architecture of iPSC. It couples a high-performance vector processor to each iPSC processing node, thus yielding a peak performance of 1,280 MFLOPS (on 32-bit data, or 424 MFLOPS on 64-bit data) on a 64-node iPSC-VX/d6 version. Each node consists of 1.5 Mbytes, giving the whole system 96 Mbytes of memory.

3.10 Massively Parallel Processor

The Massively Parallel Processor (MPP) is a bit-serial SIMD parallel processing computer built for the NASA Goddard Space Flight Center by Goodyear Aerospace Corp.. It has 16,384 processing elements (PEs).

The major blocks in the MPP are the array unit, array control unit, staging memory, program and data management unit, and interface to the host.

Logically, the array unit contains 16,384 PEs arranged in a 128 row x 128 column square array. Physically, the array unit contains an extra 128 row x 4 column rectangle of PEs for redundancy. When a faulty PE is discovered, the processor bypasses all the PEs in that column (or row), and the topology is not disturbed.

Each PE is a bit-serial processor which uses a full adder and a shift register for arithmetic. Each PE has a RAM storing 1,024 bits. The address lines of all PEs are tied together so that memories are accessed by a bit-plane with one bit of a bit-plane accessed by each PE. The PE memory can be expanded to 65,536 bits per PE or to 128 Mbytes total. The basic cycle time of the PE is 100ns. With all PEs operating in parallel, it delivers 3,000 MOPS for integer addition and 400 MFLOPS for floating point addition.

Each PE communicates with its four nearest neighbors. The edge connection is programmable. Between the top and bottom edges of the array unit, one can either connect them together to make the array look like a cylinder or separate them to make that array look like a plane. Similarly, the left and right edges can be independently connected together, or separated. When the left and right edges are connected together, one can either connect corresponding rows together or slide the connection by one row so the left PE of row i communicates with the right PE row i+1. Thus rows are connected together in a spiral fashion like a long linear string.

The array control unit has three subunits—the PE control unit (PCU) controls processing in the array unit, the I/O control unit manages flow of I/O data through the array unit, and the main control unit (MCU) runs application programs, performs necessary scalar processing, and controls the other two subunits. The division of responsibility allows array processing, scalar processing, and I/O to proceed simultaneously.

The staging memory buffers and reorders the bit-serial format of the array unit to the word format of the outside world. Data can be input at a rate of 160 Mbytes/second. An I/O rate of 320 Mbytes/second can be achieved when input and output proceed simultaneously.

The program and data management unit controls the overall flow of program and data in and out of the MPP. It also handles program development and diagnostics. It is implemented on a DEC PDP-11 minicomputer operating under DEC's RSX-11M real-time multiprogramming system.

The software consists of two assemblers (one each for the PCU and the MCU), a system subroutine library, a set of I/O macros, a control and debug module, and a linker. Additionally, a parallel version of Pascal is available.

3.11 Multimax

The Multimax multiprocessor computer system is offered by Encore Computer Corp.. The system can be configured to use from 2 to 20 processors. Its performance ranges between 1.5 and 15 MIPS.

Each Dual Processor Card (DPC) holds two independent National NS32032 processors running at 10 MHz. Each processor has its own National NS32081 floating point unit. The DCP includes a 32-Kbyte cache memory unit which minimizes processor access latency to data and instruction, and cuts bus traffic.

There are 1 to 8 Shared Memory Cards (SMCs) per system. Each card provides from 4 Mbytes to 16 Mbytes of shared memory. Thus system-wide shared memory ranges from 4 to 128 Mbytes. The SMC has two controllers, which serve the need of processors in parallel. Sequential data transfer into and out of memory is speeded up by eight-way interleaving memory bank.

A 100-Mbyte/second Nanobus, which uses advanced Schottky technology and has a cycle time of 80 ns, is

provided. It supports the full-speed operation of all processors. The Multimax vector bus can handle burst rates up to 1 million interrupts per second. All processors are available to respond to interrupts, and all processors can initiate I/O.

Multimax supports two UNIX versions—UNIX 4.2 and UNIX V, called UMAX 4.2 and UMAX V respectively. These operating systems are configured for a multiprocessing environment. They provide users with transparent parallelism in a time-sharing environment. Multimax also supplies parallel extensions to standard languages, a library of parallel processing system calls, and a parallel debugger.

3.12 NON-VON

NON-VON is a family of massively parallel computers being developed at Columbia University. The NON-VON 1 and NON-VON 3 computers are SIMD computers. The NON-VON 4 is an enhanced version designed as an ensemble of one or more independent SIMD computers communicating through a high-bandwidth interconnecting network. Thus, NON-VON 4 can operate in MSIMD modes.

All members of the family include a primary processing system (PPS), which consists of a large number of small processing elements (PEs) configured as a binary tree. Each PE has 64 bytes of local RAM and an I/O switch. The lack of memory in the PE forces it to load instructions from external sources. The design of NON-VON also includes a secondary processing system (SPS) connected to the PPS through a high-bandwidth parallel interface. The SPS can inspect records "on the fly" to determine whether they are relevant to an operation before transferring to the PPS. However, the SPS has not been implemented because of funding limitations.

NON-VON 1 and NON-VON 3 include a control processor (CP) at the root of the tree. The CP coordinates activities within the PPS and broadcasts instructions to be executed in all active PEs. On the other hand, NON-VON 4 includes a number of large processing elements (LPEs), which are capable of acting as a CP for a some portion of the PPS. Thus it can handle multitasking and multi-user applications. Additional storage is available for LPE in NON-VON 4. This will be useful as swapping storage among local RAMs in the PEs.

The NON-VON computers allow three modes of communications between the PEs. They are all supported by the I/O switch. The global communication mode allows the CP to broadcast instructions to all PEs and each PE to return results to the CP. Data can be transferred from one PE to another through the CP using global communication mode, but no concurrency is achieved. The tree communication mode allows data transfer among PEs that are physically adjacent within the PPS binary tree. Data can be transferred to the parent, left child and right child. In linear communication mode, the tree is reconfigured as a linear array of PEs. Data can be transferred to the left or right neighbor.

3.13 PASM

PASM is a partitionable SIMD/MIMD computer at Purdue University. It can be dynamically reconfigured to operate as one or more independent SIMD/MIMD computers of various sizes. The design of PASM calls for N = 1,024 processors. A 16-processor prototype based on Motorola MC68000 processors is under development.

The basic components of PASM consist of a parallel computation unit (PCU), microcontroller (MC), control storage, memory storage system, memory management system, and system control unit.

The PCU is designed to have N processors, N memory modules, and an interconnection network. The PCU processors are microprocessors that perform the actual computations. The PCU memory modules are used by the PCU processors for data storage in SIMD mode and both data and instruction storage in MIMD mode. Each memory module has a pair of memory units. This double buffering scheme allows data to be moved between one memory unit and secondary storage while the processor operates on data in the other memory unit. The interconnection network provides communication among processors.

The MC is a set of microprocessors that act as the control units for the PCU processors in SIMD mode and coordinate the activities of the PCU processors in MIMD mode. If Q MCs is the maximum number of partition allowable, then N/Q is the size of the smallest partition. Control storage contains the program for the MCs.

The memory storage system provides secondary storage for data files in SIMD mode and for data and programs files for MIMD mode. The memory management system controls the transferring of files between the memory storage system and the PCU memory modules. The system control unit is a conventional computer that coordinates different components of PASM and that is also used for program development and job scheduling.

3.14 PICAP

The PICAP is a MIMD parallel image processing system developed at the Picture Processing Laboratory, Linkoping University, Sweden. The computer has a modular architecture designed to consist of up to 16 different processors. Each processor runs independently and has a specialized function, such as linear filtering, segmentation, and image I/O. It is a multiuser system that allows users to share processors and image memory dynamically.

One of the processors is a SIMD filter processor, FIP, that consists of four subprocessors operating in parallel according to a common microprogram. Each subprocessor is pipelined to increase its performance. A 32-Kbyte local memory in the FIP is partitioned into four separate modules thus allowing simultaneous retrieval of data by the four subprocessors. For neighborhood operations, it stores a horizontal strip of an image such that pixels are both horizontally and vertically adjacent to each other. Its peak performance is 10^8 elementary operations per second on 8-bit data.

All processors operate on images stored in a large random-access image memory of four Mbytes, interleaved over 16 separate memory modules. A 40-Mbyte/second time-shared-bus connects all the processors to the memory.

The computer is programmed in three ways. One is a menu-based command language in which commands are executed directly, although it is possible to store a sequence of commands in a file. A high-level highly-interactive programming language, PPL, designed for the structure of PICAP is available. Each procedure of PPL can be executed separately. PICAP also handles programs written in FORTRAN or Pascal through a set of library routines.

3.15 RP3

Research Parallel Processor Project (RP3) was initiated at IBM. It is designed as a parallel MIMD computer for investigating both hardware and software aspects of parallel computation. It can be configured as both shared memory and local memory message-passing paradigms, as well as mixtures of the two.

RP3 is designed to have up to 512 processor/memory elements (PME) with an interconnection network. A full system is expected to provide up to 1.3 GIPS, 800 MFLOPS, 1-2 Gbytes of main storage, 192-Mbyte/second I/O rate, and 13-Gbyte/second interprocessor communication capability.

Each PME contains a 32-bit processor, 4 Mbytes of memory, a 32-Kbyte cache, a floating-point unit, and an interface to the I/O and Support Processor (ISP). The RP3 processor is a proprietary design based on the philosophy that all instructions should be completed in a single cycle. But unlike other RISC architectures, it has an extensive instruction set and performs necessary interlocks internally in hardware. The PME provides a memory mapping function as part of address translation and allows memory to be dynamically partitioned between global and local memory.

The RP3 interconnection network is composed of two networks. One provides low latency, and the other has the ability to combine messages directed to the same memory location. The low latency network is similar to an Omega network but provides dual source-sink paths.

The ISP supports I/O, monitors performance, and mediates system initialization and configuration. It is an independently programmable processor containing 4 Mbytes of memory and the same processor used in PMEs.

The operating system for RP3 will be based on BSD 4.2 UNIX. C, FORTRAN, and possibly Pascal will be available as high-level programming languages.

3.16 Warp

Warp is a MIMD one-dimensional systolic array computer being designed and built at Carniege Mellon University. It consists of ten identical cells in a linear array and delivers a peak performance of 100 MFLOPS.

Each cell contains two floating point processors, one ALU, and one multiplier. The processors are pipelined, and each can deliver up to 5 MFLOPS.

An operand register file is dedicated to each arithmetic processing units to ensure that data can be supplied at the rate they are consumed. Within each cell, a crossbar is used to support a high intra-cell bandwidth.

Every cell contains 4K of 152-bit word micro-store and 4K of 32-bit RAM as well as other registers to provide sufficient control. Each cell can be programmed individually to execute a different computation.

Data flow through the array on two data paths, while addresses and control signals travel on the address path. Each input data path has a queue to buffer input data.

An interface unit handles I/O between the array and the host and performs data conversion. Addresses for data and control signals are generated by the interface unit and are propagated from cell to cell.

Warp is designed to interface with a VAX 11/780 through an interface computer which provides 1Mbyte of memory and a 24-Mbyte/second bandwidth. The host is responsible for carrying out high-level application routines and supplying data to the Warp.

4 Further Information

Balance : Sequent Computer Systems Inc., 15450 S.W. Koll Parkway, Beaverton, OR. 97006-6063. Telephone : 503-626-5700

Butterfly : BBN Advanced Computer Inc., 10 Fawcett Street, Cambridge, MA 02238. Telephone : 617-497-3700

CAPP : Department of Computer and Information Science, University of Massachusetts, Amherst, MA 01003.

CLIP : Image Processing Group, Department of Physics and Astronomy, University College London, WC1E 6BT, United Kingdom.

Connection Machine : Thinking Computer Corporation, 245 First Street, Cambridge, MA 02142-1214. Telephone : 617-876-1111

FAIM-1 : Artificial Intelligence Laboratory, Schlumberger Palo Alto Research, 3340 Hillview Avenue, Palo Alto, CA 94304.

FX/Series : Alliant Computer Systems Corporation, 42 Nagog Park, Acton, MA 01720. Telephone : 617-263-9110

GAPP : Microelectronics Division, NCR Corp., Fort Colllins, CO.

iPSC : Intel Scientific Computers, 15201 N.W. Greenbrier Parkway, Beaverton, OR 97006. Telephone : 503-629-7600

MPP : Digital Technology Department, Goodyear Aerospace Corporation, Akron, OH 44315.

Multimax : Encore Computer Corporation, 257 Cedar Hill Street, Marlborough, MA 01752. Telephone : 617-460-0500

NON-VON : Department of Computer Science, Columbia University, New York, NY 10027.

PASM : School of Electrical Engineering, Purdue University, West Lafayette, IN 47907.

PICAP : Department of Electrical Engineering, Linkoping University, S-581 83 Linkoping University, Sweden.

RP3 : IBM T. J. Waston Research Center, Yorktown Heights, NY 10598.

Warp : Computer Science Department, Carnegie-Mellon University, Pittsburgh, PA 15213.

5 Appendix

Table of comparison

Computer	total number of processors	total processors' power	total memory size	individual processor's power	individual processor's memory
Balance	30	21 MIPS	48 Mbytes	0.7 MIPS	8 Kbytes
Butterfly	256	250 MIPS	1026 Mbytes	1 MIPS	4 Mbytes
CAPP	262,144	100 ns/cycle	1 Mbyte	100 ns/cycle	32 bits
CLIP7	2,048	100 ns/cycle	65 Kbytes	100 ns/cycle	256 bits
Connection	65,536	1,000 MIPS	32 Mbytes	0.016 MIPS	4,096 bits
FAIM-1	19	N/A	N/A	N/A	N/A
FX/8	8	94 MFLOPS	64 Mbytes	11.8 MFLOPS	8 Mbytes
GAPP	2304	921 MIPS	36 Kbytes	0.4 MIPS	128 bits
iPSC	128	1,280 MFLOPS	96 Mbytes	10 MFLOPS	4.5 Mbytes
MPP	16,384	400 KFLOPS	128 Mbytes	25 KFLOPS	65 Kbytes
Multimax	20	15 MIPS	128 Mbytes	0.7 MIPS	32 Kbytes
NON-VON	N/A	N/A	N/A	N/A	N/A
PASM	1024	N/A	N/A	N/A	N/A
PICAP	16	100 MIPS	4 Mbytes	100 MIPS	32 Kbytes
RP3	512	1.6 MFLOPS	2 Gbytes	800 MFLOPS	4 Kbytes
Warp	10	100 MFLOPS	640 Kbytes	10 MFLOPS	64 Kbytes

Table of comparison (continued)

Computer	Architecture	Market avail.	Running version	Chip used	Operating system	Programming Language
Balance	MIMD	Yes	Yes	NS32032	unix-based	C, FORTRAN, Pascal, Ada
Butterfly	MIMD	Yes	Yes	MC68020	Chrysalis	C, FORTRAN, Pascal, Lisp, Ada
CAPP	SIMD	No	No	custom	host-based	N/A
CLIP7	SIMD	No	Yes	custom	host-based	C, IPC
Connection	MSIMD	Yes	Yes	custom	host-based	C, Lisp
FAIM-1	MIMD	No	No	N/A	N/A	N/A

FX/8	MIMD	Yes	Yes	custom	unix-based	C, FORTRAN, Pascal, assembler
GAPP	SIMD	No	Yes	custom	host-based	C, assembler
iPSC	MIMD	Yes	Yes	In80286	unix-based	C, Fortran, Lisp, assembler
MPP	SIMD	No	Yes	custom	host-based	N/A
Multimax	MIMD	Yes	Yes	NS32032	unix-based	C, Fortran, Pascal, Ada, Lisp
NON-VON	MIMD	No	No	N/A	N/A	N/A
PASM	MIMD	No	No	N/A	N/A	N/A
ICAP	MSIMD	No	Yes	custom	host-based	Fortran, Pascal, assembler
RP3	MIMD	No	No	custom	unix-based	C, Fortran, Pascal
Warp	MIMD	No	Yes	custom	host-based	N/A

Evidence Combination Using Likelihood Generators

David Sher
Computer Science Department
The University of Rochester
Rochester, New York 14627

Abstract

Here, I address the problem of combining output of several detectors for the same feature of an image. I show that if the detectors return likelihoods I can robustly combine their outputs. The combination has the advantages that:

- The confidences of the operators in their own reports are taken into account. Hence if an operator is confident about the situation and the others are not, then the reports of the confident operator dominate the decision process.
- A priori confidences in the different operators can be taken into account.
- The work to combine 'N' operators is linear in 'N'.

This theory has been applied to the problem of boundary detection. Results from these tests are presented here.

1. Introduction

Often in computer vision one has a task to do such as deriving the boundaries of objects in an image or deriving the surface orientation of objects in an image. Often one also has a variety of techniques to do this task. For boundary detection there are a variety of techniques from classical edge detection literature [Ballard82] and the image segmentation literature e.g. [Ohlander79]. For determining surface orientation there are techniques that derive surface orientation from intensities [Horn70] and texture [Ikeuchi80] [Aloimonos85]. These techniques make certain assumptions about the structure of the scene that produced the data. Such techniques are only reliable when their assumptions are met. Here I show that if several algorithms return likelihoods I can derive from them the correct likelihood assuming at least one of the algorithms' assumptions are met. Thus I derive an algorithm that works well when any of the individual algorithms works well.

The mathematics here were derived independently but are similar to the treatment in [Good50]. and [Good83], using different notation. To understand my results first one must understand the meaning of likelihood.

2. Likelihoods

In this paper I call the assumptions that an algorithm makes about the world a *model*. Most models for computer vision problems describe how configurations in the real world generate observed data. Because imaging projects away information, the models do not explicitly state how to derive the configuration of the real world from the sensor data. As a result, graphics problems are considerably easier than vision problems. Programs can generate realistic images that no program can analyze.

Let O be the observed data, f a feature of the scene whose existence we are trying to determine (like a boundary between two pixels) and M a model. Many computer vision problems can be reduced to finding the probability of the feature given the model and the data, $P(f|O\&M)$. However most models for computer vision instead make it easy to compute $P(O|f\&M)$. I call $P(O|f\&M)$ (inspired by the statistical literature) the *likelihood* of f given observed data O under M. As an example assume f is "the image has a constant intensity before noise". M says that the image has a normally distributed uncorrelated (between pixels) number added to each pixel (the noise). Calculating $P(O|M\&f)$ is straight-forward (a function of the mean and variance of O).

A theorem of probability theory, *Bayes' law*, shows how to derive conditional probabilities for features from likelihoods and prior probabilities. Bayes' law is shown in equation 1.

$$P(f|O\&M)=\frac{P(O|f\&M)P(f|M)}{P(O|f\&M)P(f|M)+P(O|\sim f\&M)P(\sim f|M)} \quad (1)$$

f is the feature for which we have likelihoods. M is the domain model we are using. $P(O|f\&M)$ is the likelihood of f under M and $P(f|M)$ is the probability under M of f.

Another important use for explicit likelihoods is for use in Markov random fields. Markov random fields describe complex priors that can capture important information. Several people have applied Markov random fields to vision problems [Geman84]. Likelihoods can be used in a Markov random field formulation to derive estimates of boundary positions [Marroquin85a] [Chou87]. In [Sher86] and [Sher87] I discuss algorithms for determining likelihoods of boundaries.

Let us call an algorithm that generates likelihoods a *likelihood generator*. Consider likelihood generators L_1 and L_2 with models M_1 and M_2 and assume they both determine probability distributions for the same feature. L_1 can be considered to return the likelihood of a label l for feature f given observed data O and the domain model M_1. Thus L_1 calculates $P(O|f=l\&M_1)$. Also L_2 calculates $P(O|f=l\&M_2)$. A useful combination of L_1 and L_2 is the likelihood detector that returns the likelihoods for the case where M_1 or M_2 is true. Also the prior confidences one has in M_1 and M_2 should be taken into account.

This paper studies deriving $P(O|f=1\&(M_1 \vee M_2))$. Note that if I can derive rules for combining likelihoods for two different models then by applying the combination rules N times, N likelihoods are combined. Thus all that is needed is combination rules for two models.

3. Combining Likelihoods From Different Models

To combine likelihoods derived under M_1 and M_2 an examination of the structure and interaction of the two models is necessary. M_1 and M_2 must have the same definition for the feature being detected. If the feature is defined differently for M_1 and M_2 then M_1 and M_2 are about different events, and the likelihoods can not be combined with the techniques developed in this section.

Thus the likelihood generated by an occlusion boundary detector can not be combined with the likelihood generated by a detector for boundaries within the image of an object (such as corners internal to the image). A detector of the likelihood of heads on a coin flip can not be combined with a detector of the likelihood of rain outside using this theory. (However easy it may be using standard probability theory.)

3.1. Combining Two Likelihoods

The formula for combining the likelihoods generated under M_1 and M_2 requires prior knowledge. Necessary are the prior probabilities $P(M_1)$ and $P(M_2)$ that the domain models M_1 and M_2 are correct as well as $P(M_1\&M_2)$. Often $P(M_1\&M_2) = 0$. When this occurs the two models contradict each other. *I call two such models disjoint* because both can not describe the situation simultaneously. If M_1 is a model with noise of standard deviation is $4\pm\varepsilon$ and M_2 is a model with noise of standard deviation $8\pm\varepsilon$ then their assumptions contradict and $P(M_1\&M_2) = 0$.

Prior probabilities for the feature labels under each model ($P(f=l|M_1)$ and $P(f=l|M_2)$) are necessary. If $P(M_1\&M_2)\neq 0$ then the prior probability of the feature label under the conjunction of M_1 and M_2 ($P(f=l|M_1\&M_2)$) and the output of a likelihood generator for the conjunction of the two models ($P(O|f=l\&(M_1\&M_2))$) are needed. If I have this prior information I can derive $P(O|f=l\&(M_1 \vee M_2))$.

If I were to combine another model, M_3, with this combination I need the priors $P(M_3)$, $P(f|M_3)$, $P(M_3\&(M_1 \vee M_2)$ and $P(f|M_3\&(M_1 \vee M_2))$. To add on another mode I need another 4 priors. Thus the number of prior probabilities to combine n models is linear in n.

Thus all that is left is to derive the combination rule for likelihood generators given this prior information. The derivation starts by applying the definition of conditional probability in equation 2.

$$P(O|f=l\&(M_1 \vee M_2)) = \frac{P(O\&f=l\&(M_1 \vee M_2))}{P(f=l\&(M_1 \vee M_2))} \quad (2)$$

The formula for probability of a disjunction is applied to the numerator and denominator in equation 3.

$$P(O|f=l\&(M_1 \vee M_2)) = \frac{\left[\begin{array}{c} P(O\&f=l\&M_1) \\ + \\ P(O\&f=l\&M_2) \\ - \\ P(O\&f=l\&M_1\&M_2) \end{array}\right]}{\left[\begin{array}{c} P(f=l\&M_1) \\ + \\ P(f=l\&M_2) \\ - \\ P(f=l\&M_1\&M_2) \end{array}\right]} \quad (3)$$

In equation 4 the definition of conditional probability is applied again to the terms of the numerator and the denominator.

$$P(O|f=l\&(M_1 \vee M_2)) = \frac{\left[\begin{array}{c} P(O|f=l\&M_1)P(f=l|M_1)P(M_1) \\ + \\ P(O|f=l\&M_2)P(f=l|M_2)P(M_2) \\ - \\ P(O|f=l\&M_1\&M_2)P(f=l|M_1\&M_1)P(M_1\&M_2) \end{array}\right]}{\left[\begin{array}{c} P(f=l|M_1)P(M_1) \\ + \\ P(f=l|M_2)P(M_2) \\ - \\ P(f=l|M_1\&M_2)P(M_1\&M_2) \end{array}\right]} \quad (4)$$

Different assumptions allow different simplifications to be applied to the rule in equation 4. If the two models are disjoint equation 4 reduces to equation 5.

$$P(O|f=l\&(M_1 \vee M_2)) = \frac{\left[\begin{array}{c} P(O|f=l\&M_1)P(f=l|M_1)P(M_1) \\ + \\ P(O|f=l\&M_2)P(f=l|M_2)P(M_2) \end{array}\right]}{\left[\begin{array}{c} P(f=l|M_1)P(M_1) \\ + \\ P(f=l|M_2)P(M_2) \end{array}\right]} \quad (5)$$

Another assumption that simplifies things considerably is the assumption that prior probabilities for all feature labelings in all the models and combinations thereof are the same. I call this assumption *constancy of priors*. When constancy of priors is assumed $P(f=l|M_1) = P(f=l|M_2) = P(f=l|M_1\&M_2)$. Making this assumption reduces the number of priors that need to be determined. Since determining prior probabilities from a model is sometimes a difficult task the constancy of priors is a useful simplification. With constancy of priors equation 4 reduces to equation 6.

$$P(O|f=l\&(M_1 \vee M_2)) = \frac{\left[\begin{array}{c} P(O|f=l\&M_1)P(M_1) \\ + \\ P(O|f=l\&M_2)P(M_2) \\ - \\ P(O|f=l\&M_1\&M_2)P(M_1\&M_2) \end{array}\right]}{P(M_1)+P(M_2)-P(M_1\&M_2)} \quad (6)$$

Equation 5 with constancy of priors reduces to equation 7.

$$P(O|f=l\&(M_1 \vee M_2)) = \frac{\left[\begin{array}{c} P(O|f=l\&M_1)P(M_1) \\ + \\ P(O|f=l\&M_2)P(M_2) \end{array}\right]}{P(M_1)+P(M_2)} \quad (7)$$

Thus equation 7 describes the likelihood combination rule with disjoint models and constancy of priors.

3.2. Understanding the Likelihood Combination Rule

The easiest incarnation of the likelihood combination rule to understand is the rule for combining likelihoods from disjoint models given constancy of priors across models (equation 7). Here the combined likelihood is the weighted average of the likelihoods from the individual models weighted by the probabilities of the models applying. (The combined likelihood is the likelihood given the disjunction of the models).

If models M_1 and M_2 are considered equally probable and the likelihoods returned by M_1's detector are considerably larger than those of M_2's detector then the probabilities determined from the combination of M_1 and M_2 are close to those determined from M_1. Thus a model with large likelihoods determines the probabilities. To illustrate this principle consider an example.

Assume that a coin has been flipped $n+1$ times. The results of flipping it has been reported for the first n times. The task is to determine the probability of heads having been the result of the $n+1^{st}$ flip. Consider the results of each coin flip independent. Let M_1 be the coin being fair so that the probability of heads and tails is equal. Let M_2 be that the coin is biased with the probability of heads is π and tails $1-\pi$ with π being a random choice with equal probability between p and $1-p$. Hence the coin is biased towards heads or tails with equal probability but the bias is consistent between coin tosses. The probability of heads remains the same for all coin tosses in both models. M_1 and M_2 are disjoint (the coin is either fair or it isn't but not both) and the prior probability of a flip being heads or tail is the same for both, .5.

Under M_1 the probability of each of the possible flips of $n+1$ coins is 2^{-n-1}. Under M_2 the probability of $n+1$ flips of coins with h heads and $t=n+1-h$ tails is:

$$\tfrac{1}{2}p^h(1-p)^t+\tfrac{1}{2}p^t(1-p)^h$$

Let $n=2$ and $p=.9$. Assume the first two flips are both heads. Let H be "the third flip was heads" and T be "the third flip was tails." The likelihood of H given the observed data is the probability of all 3 flips being heads divided by the probability of the third flip being heads. The likelihood of T given the observed data is the probability of the first 2 being heads and the 3rd tails divided by the probability of the third flip being tails.

Under M_1 the probability of all 3 flips being heads is 0.125 and the probability of a flip being heads is 0.5 thus the likelihood of H is 0.25. The likelihood of T is 0.25 by the same reasoning. Applying Bayes' law to get the probability of H under M_1 one derives a probability of .5 .

Under M_2 the probability of all 3 flips being heads is 0.365 and the probability of a flip being heads is 0.5. Thus the likelihood of H is 0.73. Under M_2 the probability of the first two being heads and the third being tails is 0.045 and the probability of a flip being tails is 0.5. Thus the likelihood of T is 0.09. Applying Bayes' law under M_2 a probability of H being 0.89 is derived.

If M_1 and M_2 are considered equally probable then the combination of the likelihoods from the two models is the average of the two likelihoods. Thus the likelihood of H for this combination is 0.49 and the likelihood of T is 0.17 (likelihoods don't have to sum to 1). Bayes' law combines these probabilities to get 0.74 for the 3rd flip to be heads.

The table in figure 1 describes combining various M_2's with different values of p with M_1 for the different combinations with $n=4$

Look at the probabilities with $p=.9$ and the observed data is HHHH. For this case the observed data fits M_2 much better than M_1 and the probability from combining M_1 and M_2 is close to the probability resulting from using just M_2, .9. If we had a longer run of heads the probability of future heads would approach exactly M_2's prediction, .9. On the other hand if we had a long run of equal numbers of heads and tails the probability of future heads would quickly approach the prediction of M_1, .5. When the observed data is HHHT the observed data fits M_1 about as well as M_2 and the resulting probability is near the average of .5 predicted by M_1 and 0.8902 predicted by M_2. Thus when the observed data is a good fit for a particular model (like M_2) the probabilities predicted by the combination is close to the probabilities predicted by the fitted model. If two models fit about equally then the result is an average of the probabilities[1].

4. Results

I have applied this evidence combination to the boundary detection likelihood generators described in [Sher87]. Here I prove my claims that the evidence combination theory allows me to take a set of algorithms that are effective but not robust and derive an algorithm that is robust. The output of such an algorithm is almost as good as the best of its constituents (the algorithms that are combined).

4.1. Artificial Images

Artificial images were used to test the algorithms described in section 3 quantitatively. I used as a source of likelihoods the routines described in [Sher87]. Because the positions of the boundaries in an artificial image are known one can accurately measure false positive and negative rates for different operators. Also one can construct artificial images to precise specifications. The artificial images I use is an image composed of overlapping circles with constant intensity and aliasing at the boundaries shown in figure 2.

[1]However the feature that the decision theory predicts is not the average of the features predicted under the two different models in general.

Observed Coin Flips	Combined with M_1 or just M_2	Likelihood of H		Likelihood of T		Probability of H	
		$p=.6$	$p=.9$	$p=.6$	$p=.9$	$p=.6$	$p=.9$
HHHH	Just M_2	0.088	0.5905	0.0672	0.0657	0.567	0.8999
	Combined	0.07525	0.3265	0.06485	0.0641	0.537	0.8359
HHHT	Just M_2	0.0672	0.0657	0.0576	0.0081	0.5385	0.8902
	Combined	0.06485	0.0641	0.06005	0.0353	0.5192	0.6449
HHTT	Just M_2	0.0576	0.0081	0.0575	0.0081	0.5	0.5
	Combined	0.06005	0.0353	0.06	0.0353	0.5	0.5
HTTT	Just M_2	0.0576	0.0081	0.0672	0.0657	0.4615	0.1098
	Combined	0.06005	0.0353	0.06485	0.0641	0.4808	0.3551
TTTT	Just M_2	0.0672	0.0657	0.088	0.5905	0.433	0.1001
	Combined	0.06485	0.0641	0.07525	0.3265	0.4629	0.1641

Figure 1: Result of likelihood combination Rule

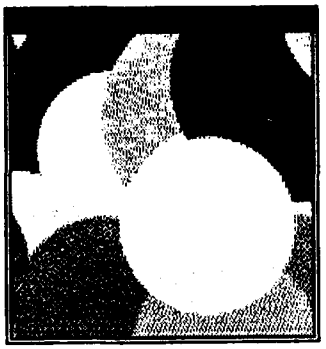
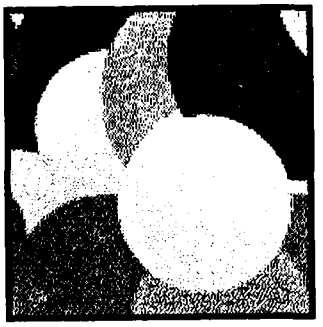
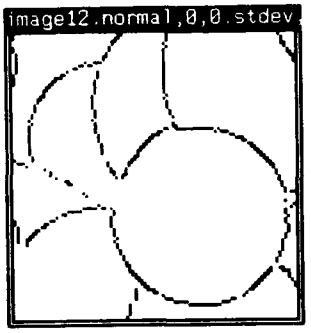


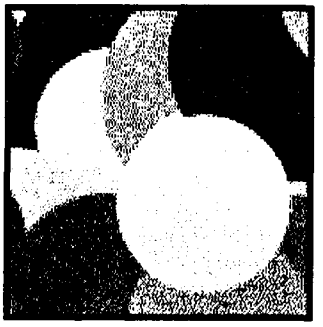
Figure 2: Artificial Test Image

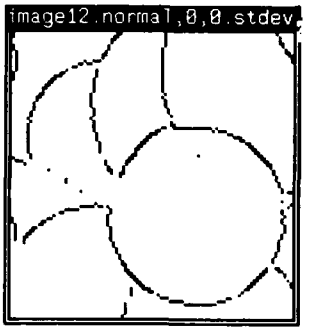


a: Image with stdev 12 noise b: Output of stdev 4 detector
Figure 3: Stdev 4 detector applied to 2 image with stdev 12 noise



a: Image with stdev 12 noise b: Output of stdev 12 detector
Figure 4: Stdev 12 detector applied to 2 image with stdev 12 noise





a: Image with stdev 12 noise b: Output of combined detector
Figure 5: Combined detector applied to 2 image with stdev 12 noise

The intensities of the circles were selected from a uniform distribution from 0 to 254. To the circles were added normally distributed uncorrelated noise with standard deviations 4, 8, 12, 16, 20, and 32. The software to generate images of this form was built by Myra Van Inwegen working under my direction. This software will be described in an upcoming technical report.

In figure 3 I show the result of applying the detector tuned to standard deviation 4 noise to the artificial image with standard deviation 12 noise added to it. In figure 4 I show the result of applying the detector tuned to standard deviation 12 noise to an image with standard deviation 12 noise added to it. In figure 5 I show the result of applying the combination of the detectors tuned to 4, 8, 12, and 16 standard deviation noise. The combination rule was that for disjoint models with the same priors. The 4 models were combined with equal probability. These operator outputs are thresholded at 0.5 probability with black indicating an edge and white indicating no edge.

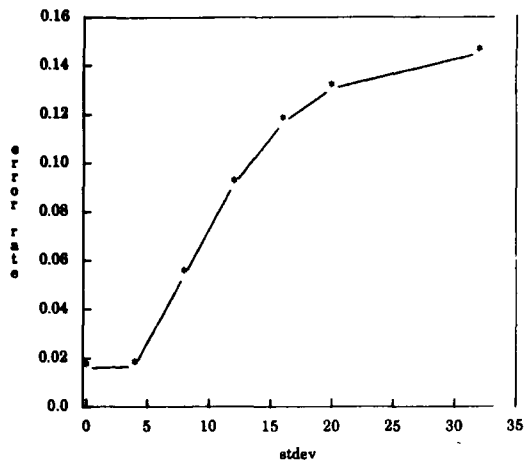


Figure 6: Total errors by the stdev 4 detector

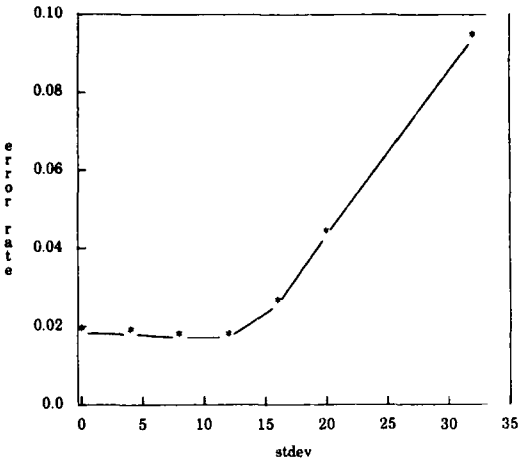


Figure 7: Total errors by the stdev 12 detector

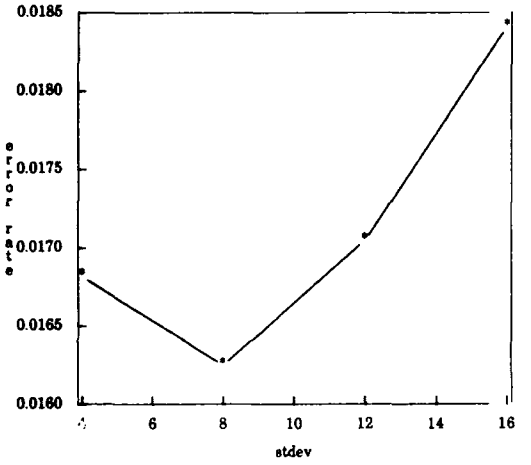


Figure 8: Total errors by the tuned detector

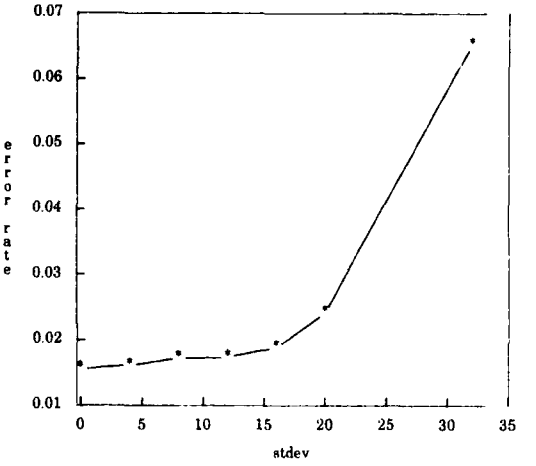
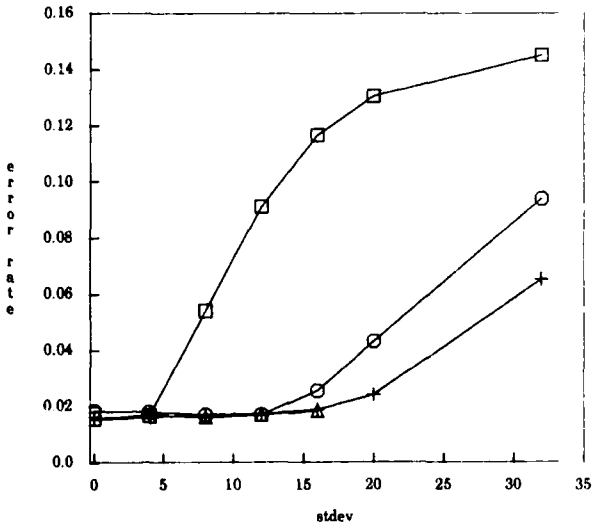


Figure 9: Total errors by the combined detector

Note that the result of using the combined operator is similar to that of the operator tuned to the correct noise level. Most of the false boundaries found by the stdev 4 operator are ignored by the combined operator.

Using this artificial image I have acquired statistics about the behavior of the combined detector vs the tuned ones under varying levels of noise. I have charted the total error rate for the artificial image in figure 2 with normally distributed spacially independent noise with increasing standard deviation. Figure 6 shows the error rate for a detector tuned to standard deviation 4 noise as the noise in the image increases. Figure 7 shows the error rate for a standard deviation 12 operator. Figure 8 shows the error rate for an operator tuned to the current standard deviation of the noise. Figure 9 shows the error rate of the detector that is the combination of the detectors tuned to a standard deviation of 4, 8, 12 and 16 using the rule for combining disjoint models witht the same priors for the feature (since the probability of a boundary is unaffected by the noise level of the sensor). Figure 10 shows the superposition of the 4 previous graphs.



square: stdev 4 operator circle: stdev 12 operator
triangle: tuned operator cross: combined operator

Figure 10: Total errors by all detectors

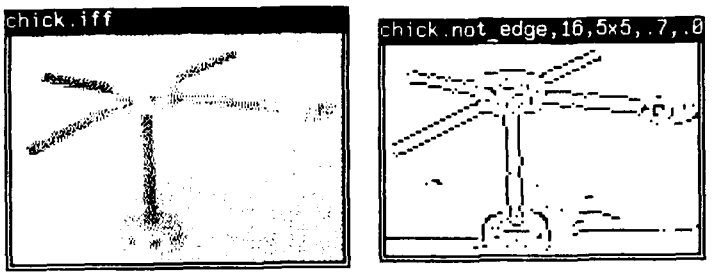
The error rate for the combined operator is always nearly as low as that of the tuned operator (shown as triangles). It is lower than the standard deviation 12 operator when the noise is low and lower than the standard devation 4 operator when the noise is high. These results are evidence that my combination rule is robust.

4.2. Real Images

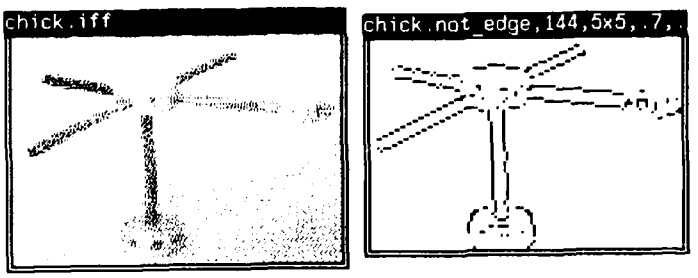
I have also tested these theories using two images taken by cameras. One of these images is a tinker toy image taken in our lab. The other is an aerial image of the vicinity of Lake Ontario. Figure 11 shows the result of the operator tuned to standard deviation 4 noise applied to the tinker toy image and thresholded at 0.5 probability. Figure 12 shows the result of the operator tuned to standard deviation 12 noise applied to the tinker toy image. Figure 13 shows the effect of combining operators tuned to standard deviation 4, 8, 12 and 16 with equal probability.

Here, the result of the combined operator seems to be a cleaned up version of the standard deviation 4 operator. Most of the features that are represented in the output of the combined operator are however real features of the scene. The line runing horizontally across the image that the standard deviation 4 operator and the combined operator found is the place where the table meets the curtain behind the tinkertoy. The standard deviation 4 operator was certain of its interpretation and the other operators were uncertain at that point so its interpretation was used by the combination.

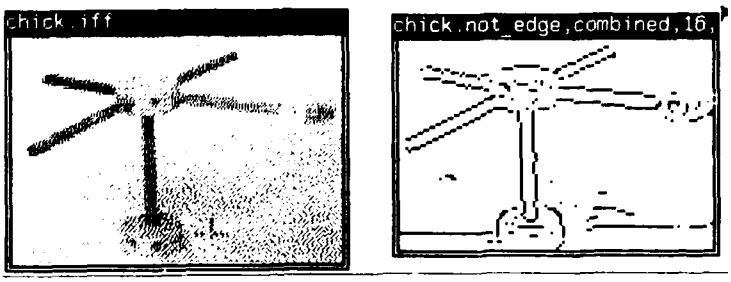
The results from the aerial image are also instructive. Figure 14 shows the result of the operator tuned to standard deviation 4 noise applied to the aerial image and thresholded at 0.5 probability. Figure 15 shows the result of the operator tuned to standard deviation 12 noise applied to the aerial image. Figure 16 shows the effect of combining operators tuned to standard deviation 4, 8, 12 and 16 with equal probability.



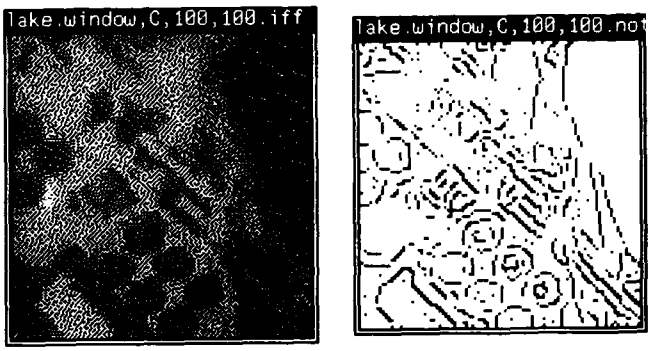
a: Tinkertoy Image b: Output of stdev 4 detector
Figure 11: Stdev 4 detector applied to tinkertoy image



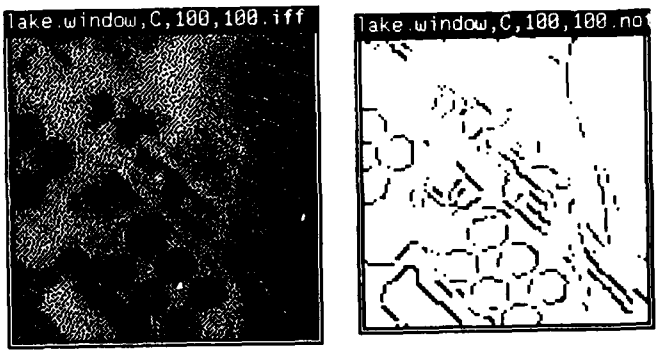
a: Tinkertoy Image b: Output of stdev 12 detector
Figure 12: Stdev 12 detector applied to tinkertoy image



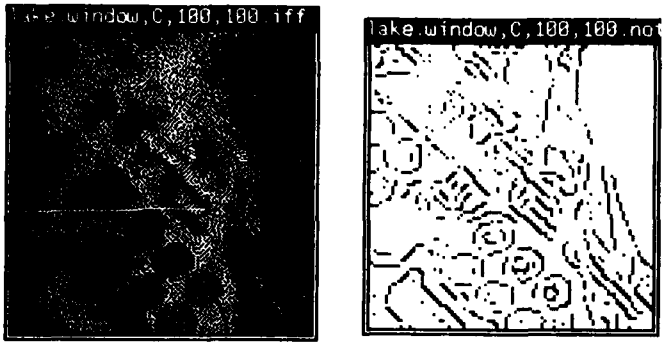
a: Tinkertoy Image b: Output of combined detector
Figure 13: Combined detector applied to tinkertoy image



a: Aerial Image b: Output of stdev 4 detector
Figure 14: Stdev 4 detector applied to aerial image



a: Aerial Image b: Output of stdev 12 detector
Figure 15: Stdev 12 detector applied to aerial image



a: Aerial Image b: Output of combined detector

Figure 16: Combined detector applied to aerial image

The results from the combined operator are again a cleaned up version of the results from the standard deviation 4 operator. I believe this behavior occurs again because the features being found by the standard deviation 4 operator are in the scene. However I do not have the intimate knowledge of the aerial image that I do of the tinkertoy image.

4.3. Future Experiments

Soon, I will apply my evidence combination rules to operators that make different assumptions about the expected image histogram. The operator used so far in my experiments expects a uniform histogram between 0 and 254. Currently, a likelihood generator has been built that assumes a triangular distribution with the probability of an object having intensity less than 128 being one fourth the probability of an object having intensity greater than or equal to 128. It is not clear that the probabilities calculated based on this assumption will be significantly different from those based on the uniform histogram assumption. If there is no difference in the output of two operators the effect of combination is invisible.

Larger operators will soon be available. The likelihoods generated based on these larger operators would be finely tuned. The same evidence combination can be applied to these operators.

Likelihoods are used by Markov random field algorithms to determine posterior probabilities [Marroquin85a] [Chou87]. Likelihoods resulting from my combination rules can be used by Markov random field algorithms.

5. Previous Work

Much of the work on evidence and evidence combination and vision has been on high level vision. An important Bayesian approach (and a motivation for my work) was by Feldman and Yakimovsky [Feldman74]. In this work Feldman and Yakimovsky were studying region merging based on high level constraints. They first tried to find a probability distribution over the labels of a region using its characteristics such as mean color or texture. They then tried to improve these distributions using labelings for the neighbors. Then they made merge decisions based on whether it was sufficiently probable that two adjacent regions were the same.

Work with a similar flavor has been done by Hanson and Riseman. In [Hanson80] Bayesian theories are applied to edge relaxation. This work had serious problems with its models and the fact that the initial probabilities input were edge strengths normalized never to exceed 1. Of course such edge strengths have little relationship to probabilities (a good edge detector tries to be monotonic in its output with probability but that is about as far as it gets). In [Wesley82a] and [Wesley82b] Dempster-Shafer evidence theory is used to model and understand high level problems in vision especially region labeling. In [Wesley82b] there is some informed criticism of Bayesian approaches. In [Reynolds85] They study how one converts low level feature values into input for a Dempster-Shafer evidence system.

There has been much use of likelihoods in recent vision work. In particular work based on Markov random fields [Geman84] [Marroquin85b] [Marroquin85a] use likelihoods. A Markov random field is a prior probability distribution for some feature of an image and the likelihoods are used to compute the marginal posterior probabilities that are used to update the field. Haralick has mentioned that his facet model [Haralick84] [Haralick86b] can be easily used to build edge detectors that return likelihoods [Haralick86a]. I also have built boundary detectors that return likelihoods and the results of using them is documented in [Sher87]. Paul Chou is using the likelihoods I produce with Markov random fields for edge relaxation [Chou87]. He is also studying the use of likelihoods for information fusion. Currently, he is concentrating on information fusion from different sources of information.

6. Conclusion

I have presented a Bayesian technique for information fusion. I show how to fuse information from detectors with different models. I presented results from applying these techniques to artificial and real images.

These techniques take several operators that are tuned to work well when the scene has certain particular properties and get an algorithm that works almost as well as the best of the operators being combined. Since most algorithms available for machine vision are erratic when their assumptions are violated this work can be used to improve the robustness of many algorithms.

7. Acknowledgements

This work would have been impossible without the advice and argumentation of such people as Paul Chou and Mike Swain (who has made suggestions from the beginning) and of course my advisor Chris Brown, and who could forget Jerry Feldman. This work was supported by the Defense Advanced Research Projects Agency U. S. Army Engineering Topographic Labs under grant number DACA76-85-C-0001, the National Science Foundation under grant number DCR-8320136. Also this work ws supported by the Air Force Systems Command, Rome Air Development Center, Griffiss

Air Force Base, New York 13441-5700, and the Air Force Office of Scientific Research, Bolling AFB, DC 20332, under Contract No. F30602-85-C-0008. This contract supports the Northeast Artificial Intelligence Consortium (NAIC).

References

[Aloimonos85]
J. Aloimonos and P. Chou, Detection of Surface Orientation and Motion from Texture: 1. The Case of Planes, 161, Computer Science Department, University of Rochester, January 1985.

[Ballard82]
D. H. Ballard and C. M. Brown, in *Computer Vision*, Prentice-Hall Inc., Englewood Cliffs, New Jersey, 1982, 125.

[Chou87] P. Chou, Multi-Modal Segmentation using Markov Random Fields, *Submitted to IJCAI*, January 1987.

[Feldman74]
J. A. Feldman and Y. Yakimovsky, Decision Theory and Artificial Intelligence: I. A Semantics-Based region Analyzer, *Artificial Intelligence 5*(1974), 349-371, North-Holland Publishing Company.

[Geman84]
S. Geman and D. Geman, Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images, *PAMI 6*,6 (Novermber 1984), 721-741, IEEE.

[Good50] I. J. Good, *Probability and the Weighing of Evidence*, Hafner Publishing Company., London, New York, 1950

[Good83] I. J. Good, Subjective Probability as the Measure of a Non-measurable Set, in *Good Thinking: The Foundations of Probability and its Applications*, Minneapolis (editor), University of Minnesota Press, Minneapolis, 1983, 73-82.

[Hanson80]
A. R. Hanson, E. M. Riseman and F. C. Glazer, Edge Relaxation and Boundary Continuity, 80-11, University of Massachusetts at Amherst, Computer and Information Science, May 1980.

[Haralick84]
R. M. Haralick, Digital Step Edges from Zero Crossing of Second Directional Derivatives, *PAMI 6*,1 (January 1984), 58-68 IEEE.

[Haralick86a]
R. Haralick, Personal Communication, June 1986.

[Haralick86b]
R. M. Haralick, The Facet Approach to Gradient Edge Detection, *Tutorial 1 Facet Model Image Processing (CVPR)*, May 1986.

[Horn70] B. K. P. Horn, *Shape from Shading: A Method for FInding the SHape of a Smooth Opaque Object from One View*, Massachusetss Institute of Technology Department of ELectrical Engineering., August 1970

[Ikeuchi80]
K. Ikeuchi, Shape form Regular Patterns (an Example of Constraint Propagation in Vision), 567, Massachusetts Institute of Technology, Artificial Intelligence Laboratory, March 1980.

[Marroquin85a]
J. L. Marroquin, Probabilistic Solution of Inverse Problems, Tech. Rep. 860, MIT Artificial Intelligence Laboratory, September 1985.

[Marroquin85b]
J. Marroquin, S. Mitter and T. Poggio, Probabilistic Solution of Ill-Posed Problems in Computational Vision, *Proceedings: Image Understanding Workshop*, December 1985, 293-309. Sponsored by: Information Processing Techniques Office Defence Advanced Research Projects Agency.

[Ohlander79]
R. Ohlander, K. Price and D. R. Reddy, Picture Segmentation using a Recursive Region Splitting Method, *CGIP 8*,3 (1979).

[Reynolds85]
G. Reynolds, D. Strahman and N. Lehrer, Converting Feature Values to Evidence, *PROCEEDINGS: IMAGE UNDERSTANDING WORKSHOP*, December 1985, 331-339. Sponsored by: Information Processing Techniques Office, Defence Advanced Research Projects Agency.

[Sher86] D. Sher, Optimal Likelihood Detectors for Boundary Detection Under Gaussian Additive Noise, *IEEE Conference on Computer Vision and Pattern Recognition*, Miami, Florida, June 1986.

[Sher87] D. B. Sher, Advanced Likelihood Generators for Boundary Detection, TR197, University of Rochester Computer Science Department, Rochester NY 14627, January 1987.

[Wesley82a]
L. P. Wesley and A. R. Hanson, The Use of an Evidential-Based Model for Representing Knowledge and Reasoning about Images in the Visions System, *PAMI 4*,5 (Sept 1982), 14-25, IEEE.

[Wesley82b]
L. P. Wesley and A. R. Hanson, The use of an Evidential-Based Model for Representing Knowledge and Reasoning about Images in the VISIONS System, *Proceedings of the Worksshop on Computer VIsion: Representation and Control*, August 1982, 14-25.

Multi-Modal Segmentation using Markov Random Fields

Paul B. Chou and Christopher M. Brown
Computer Science Department
The University of Rochester
Rochester, New York 14627
ARPA: chou@rochester.arpa brown@rochester.arpa
UUCP: {allegra,seismo,decvax}!rochester!chou
{allegra,seismo,decvax}!rochester!brown

ABSTRACT

This paper develops a new approach, based on Bayesian probability theory, to combine information from various sources for image segmentation. In this approach, the observable evidence and prior knowledge are separately modeled due to their distinct characteristics. A set of early visual modules provide opinions about individual image elements based on disparate sources of image observations. Their opinions are combined coherently and consistently through a hierarchically structured knowledge tree. The prior knowledge about spatial interactions of the image features is modeled by using Markov Random Fields. This approach constantly maintains the *a posteriori* probabilities of sementations resulting from combining the prior knowledge and the available opinions. The probabilistic justification for this approach is provided. A set of experimental results using synthetic input and a stochastic relaxation estimation procedure demonstrates some of the advantages of this approach.

0. Background and Motivation

An important aspect of computer vision is to infer a representation of a three dimensional scene from two dimensional projections yielding features like shadows, stereo disparities, texture, and optical flow - all corrupted by noise. The problem is ill-posed given only the input projections and the characteristics of the imaging devices. Therefore, computer vision research does not just study the geometry and the photometry of the cameras, but also relies on scene modeling (prior knowledge) and mechanisms for integrating the scene models with the input observations to make 3D inferences possible.

A major portion of computer vision research for the past decade has been devoted to the so-called intrinsic image computations [Barrow & Tenenbaum, 78][Marr, 82]. The emphasis of these efforts is on the relations between scene parameters and individual image features (e.g. shape-from-shading and shape-from-contours). Due to the ill-posed nature of each problem, most of them assume that the images have been already segmented into "homogeneous" regions so that they can apply some global assumptions (prior knowledge) to regularize their computations [Ikeuchi and Horn, 81] [Grimson, 81] [Terzopoulos, 86]. The global assumptions imposed by different computational modules imposed may be inconsistent, so it is not obvious how to combine their results coherently.

We propose a new approach, based on Bayesian probability theory, for integrating information from various sources. It consistently and coherently combines the opinions of the early modules and updates the probabilities of the image features accordingly. Each of the early visual modules acts as an expert on a set of hypotheses in a hierarchical knowledge structure H that relates individual image elements. The prior knowledge about the spatial interactions of the image features is modeled as Markov Random Fields. This approach is useful for the image segmentation problem, and can be extended to solve other signal estimation problems as well.

The Bayesian formalism is widely used as a tool for updating "belief" as new information is acknowledged. Bolles uses a set of probabilistic feature detectors to look for certain objects in the image [Bolles, 77]. Witkin uses the Bayesian estimation paradigm to estimate the orientation of a textured plane [Witkin, 81]. [Feldman and Yakimovsky, 74] [Geman and Geman, 84] [Marroquin, 85] [Elliott and Derin, 84] use it to solve the segmentation problem. In [Bolle and Cooper, 84], a Bayesian classifier is developed to estimate the underlying surface types of image regions. A class of research known as Probabilistic Relaxation [Rosenfeld *et al*, 76] [Shvayster and Peleg, 85] [Hummel and Zucker, 83], was also inspired by the Bayesian formalism. It interprets spatial knowledge as statistical quantities related to the conditional probabilities, correlations, or compatibility among neighboring elements, and develops updating rules to incorporate contextual information.

What is lacking in the previous work, however, is a uniform fusion mechanism to integrate disparate visual information in a probabilistically justifiable manner.

A visual module could provide opinions on any mutually exclusive set of labels in the hierarchical knowledge tree H. We shall develop an evidence aggregation method that combines consistently and coherently the opinions of the

visual modules on a label tree. This method, based on the reasoning proposed in [Pearl, 86], follows the Bayesian formalism. It does not require any knowledge of prior probabilities. It requires only trivial computations. With this method, we are able to design individual experts for a subset of labels of the tree without having to know about the rest of the world. The combined opinion can thus be fused with the image knowledge represented by the *a priori* probabilistic distributions.

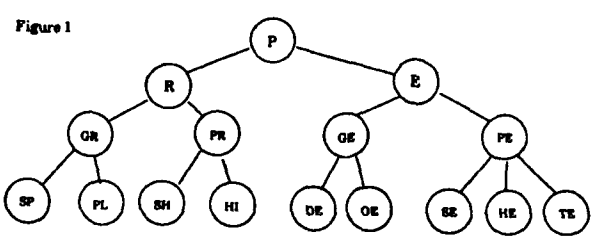
In Section 1, we define the segmentation problem and describe the representations for prior knowledge and the opinions from early modules. In Section 2, we discuss how to encode the prior knowledge as Markov Random Fields. The method for combining opinions of the early modules and its probabilistic justification are presented in Section 3. Section 4 shows the computations for the *a posteriori* probabilities. Illustrative experimental results are shown in Section 5.

1. A Probabilistic View of the Image Segmentation Problem

Represent an image as a set of primitive elements $S = \{s_1, s_2, \ldots, s_N\}$. A segmentation ω of the image with respect to a label set $L = \{l_1, l_2, \ldots, l_Q\}$ is a mapping from S to L. Let $\omega_i = \omega(s_i) \in L$ represent the label attached to s_i in segmentation ω. Let Ω be the set of all segmentations. The image segmentation problem with respect to L can be loosely described as to find the $\omega \in \Omega$ that "best fits" the information collected subject to the limitation of the computational resources. This section describes the uses of probability as the representation for various kinds of information and the corresponding criteria for finding the "best fit".

It is frequently desirable to organize segmentation labels as a hierarchical tree. Figure 1 shows one example of such trees. Each internal node in a tree of labels represents the disjunction of its sons. Each cross-section is a mutually exclusive and exhaustive label set; i.e., a segmentation problem can be defined with respect to a cross-section in a label tree. Using such a tree, we can represent a particular piece of knowledge about the labels at whatever level of abstraction that is appropriate. For the rest of the paper, we use L to denote a set of mutually exclusive and exhaustive set of labels in a label tree H.

Figure 1



An example of label trees

The semantics of the labels: P - primitive element, E - edge, R - region, PE - photometrical edge, GE - geometrical edge, PR - photometrical region, GR - geometrical region, TE - texture edge, HE - highlight edge, SE - shadow edge, OE - orientation edge, DE - depth edge, HI - highlight region, SH - shadow region, PL - planar region, SP - spherical region.

1.1. Global Prior Knowledge

Let $X = \{X_s, s \in S\}$ be a set of random variables indexed by S, with $X_s \in L$ for all s. A segmentation can be considered a realization, or a *configuration*, of this random field and Ω can be considered the configuration space of X. Ideally the prior knowledge about Ω can be represented by a probability distribution over Ω. In practice this distribution is either unobtainable or unmanageable due to the immense size (Q^N) of the sample space. In many image understanding applications, however, some restricted classes of distributions can model the image adequately due to the local behavior of the image phenomena. In this paper, we will exclusively use Markov Random Fields (MRFs) as the *a priori* models for Ω, but the work illustrated here can be extended to other image models as well. Section 2 will discuss the MRF model in detail.

1.2. Local Visual Observations

We model early visual computations as the computations performed by a set of independent modules. The input for these modules is noise-corrupted visual data, such as image irradiance, texture, stereo disparities, etc. When making an opinion about image element s, a module may restrict its consideration of input to some spatial region dependent on s. Typically, the region will include s and its spatially adjacent elements. Also, a module might not use all the input available for a given element, that is, it might use only irradiance when other data are also available. The subset of input used to make an opinion about element s is observation O_s.

In our treatment, the *opinions* of the modules are presented in terms of likelihood ratios. For example, module A is an expert on the label set $L_A = \{l_i, l_j\} \subseteq L$. After observing O_s, the module reports one likelihood ratio for each label in L_A. A *likelihood ratio* is the probability of the observation given that one label truly applies divided by the probability of the observation should none of the labels in L_A apply. For example, the likelihood ratio reported for label l_i is:

$$\lambda_i^A = \frac{P(O_s|l_i)}{P(O_s|\neg(\bigcup_{l \in L_A} l))} \tag{1.0}$$

Note that we define $P(O|\neg(\bigcup_{l \in L_A} l))$ to be 1 when L_A is exhaustive. The methods for designing such modules are well known. Interested readers can consult [Bolles, 77] and [Sher, 87].

For a purpose that will soon become clear, we impose the following assumption of conditional independence between spatially distinct observations:

$$P(O^A|\omega) = \prod_{s \in S} P(O_s^A|\omega_s) \tag{1.1}$$

where the superscript A indicates the observations of the module A. This assumption has been used implicitly in numerous applications and is valid whenever the noise processes are spatially independent [Derin and Cole, 86][Marroquin, 85].

1.3. Posterior Probability and Bayesian Estimation

Following the Bayesian formalism, the goodness of a segmentation can be evaluated in terms of its *a posteriori* expected loss,

$$E(Loss(\omega|O)) = \sum_{f\in\Omega} loss(\omega,f)P(f|O) \qquad (1.2)$$

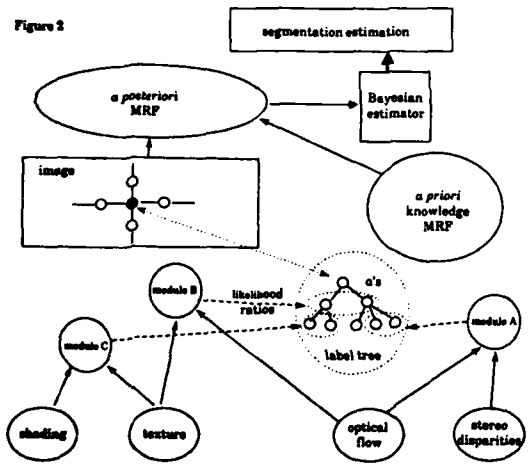
where the $P(f|O)$ denotes the posterior probability of f given the observation O. Bayes' rule can then be used to derive the *a posteriori* probability

$$P(\omega|O) = \frac{P(\omega)P(O|\omega)}{\sum_{f\in\Omega}P(f)P(O|f)} \qquad (1.3)$$

From (1.1), observe that scaling all $P(O_s|l)$ by a constant factor for fixed s does not change the posterior distribution in (1.3). This fact allows us to combine the likelihoods without having to normalize the results.

The choice for the loss function depends on the characteristics of a particular application. Different estimation algorithms can be designed according to the different loss functions. In [Geman and Geman, 84], the Maximum A Posteriori (MAP) estimation is used. A simulated annealing procedure with a stochastic sampler (Gibbs sampler) carries out the computation. In [Marroquin, 85], the Maximizer of the Posterior Marginals (MPM) estimation is proposed for the segmentation problem. A Monte Carlo procedure that uses the Gibbs sampler collects sample statistics and makes the estimation. Figure 2 is a block diagram for a segmentation system following the methodology proposed in this paper. This approach, computing the *a posteriori* probabilities for the segmentations given the set of opinions from the early modules and the *a priori* probability distribution of the image, can support both the MAP and MPM estimation methods as well as other Bayesian estimations.

Figure 2



2. Markov Random Fields

Markov Random Fields have been used for image modeling in many applications for the past few years [Hassner and Slansky, 80] [Geman and Geman, 84] [Marroquin, 85] [Cross and Jain, 83] [Derin and Cole, 86]. One of the most successful applications of MRFs is to model the spatial interactions of image features. In this section, we review the properties of MRFs and describe how to encode prior knowledge in this formalism. We refer the reader to [Kinderman and Snell, 81] for an extensive treatment of MRFs.

2.1. Definition

Let $X = \{X_s, s\in S\}$ be a set of random variables indexed by S and E a set of unordered 2-tuple (s_i, s_j)'s representing the connections between the elements in S. The set E defines a neighborhood system $N = \{N_s|s\in S\}$, where N_s is the neighborhood of s in the sense that

(1) $s\notin N_s$, and

(2) $r\in N_s$ if and only if $(s, r)\in E$.

Let $\omega = \{X_s = \omega_s, s\in S\}$ be a configuration of X, $\omega_s\in L$. and Ω the set of all possible configurations. We say X is a *Markov Random Field* with respect to N if and only if

$$P(X=\omega) > 0 \text{ for all } \omega\in\Omega \qquad (2.1)$$

$$P(X_s=\omega_s|X_r=\omega_r, r\in S, r\neq s) = P(X_s=\omega_s|X_r=\omega_r, r\in N_s) \qquad (2.2)$$

The conditional probabilities in the right-hand side of (2.2) are called the local characteristics that characterize the random field. An intuitive interpretation of (2.2) is that the contextual information provided by $S-s$ to s is the same as the information provided by the neighbors of s. Thus the effects of members of the field upon each other is limited to local interaction as defined by the neighborhood. A very desirable property of MRFs that makes them attractive to scientists in many disciplines is the MRF-Gibbs equivalence described in the following theorem.

2.2. MRF-Gibbs Equivalence

Hammersley-Clifford Theorem: A random field X is an MRF with respect to the neighborhood system N if and only if

$$P(\omega) = \frac{e^{-\frac{1}{T}U(\omega)}}{Z} \quad \text{for all } \omega\in\Omega \qquad (2.3)$$

where

$$U(\omega) = \sum_{c\in C} V_c(\omega) \qquad (2.4)$$

C is the set of totally connected subgraphs (cliques) with respect to N. Z is a normalizing constant, so that the probabilities of all realizations sum to one.

Several terminologies from Physics can provide intuition about the Gibbs measure - the right-hand side of (2.3). T is the *temperature* of the field that controls the flatness of the distribution of the configurations. A *potential* V is a way to assign a number $V_c(\omega)$ to every subconfiguration ω_c of a

configuration ω, where $c \in C$. $U(\omega)$, the sum of the local potentials, is the *energy* of the configuration ω. A system is in *thermal equilibrium* when the probabilities of its configurations follows the Gibbs measure.

With the Hammersley-Clifford Theorem, a MRF can be characterized by the potential function V instead of the local characteristics. The joint probabilities of the random variables in X can be computed by summing up the local potential assignments. The local characteristics can be computed from the potential function through the following relation:

$$P(X_s = \omega_s | X_r = \omega_r, r \in N_s) = \frac{e^{-\frac{1}{T}\sum_{c \in C_s} V_c(\omega)}}{\sum_{\omega'} e^{-\frac{1}{T}\sum_{c \in C_s} V_c(\omega')}} \qquad (2.5)$$

where C_s is the set of cliques that contain s, and ω' is any configuration of the field that agrees with ω everywhere except possibly s. [Geman and Geman, 84] proves that by using the Gibbs sampler - randomly and repeatedly selecting configurations for each variable in X according to (2.5), the field will reach thermal equilibrium eventually.

2.3. Encoding Prior Knowledge

For the image segmentation problem, the goal is to choose an appropriate neighborhood system and a potential function for the random field X over the image S to represent our prior knowledge about the image. The neighborhoods should be large enough to capture the interactions between the primitive elements but still small enough for a machine to carry out the computations required to make an estimation. For a lattice-structured image, four and eight-connected neighborhood systems are the most commonly used. Once a neighborhood system has been decided ; the potential function can be estimated, at least in principle, given a set of realizations of the uncorrupted image. Some pioneer work has studied several statistical estimation methods for homogeneous MRFs over lattices [Cross and Jain, 83] [Elliott and Derin, 84]. We feel that a general theory for choosing the neighborhood system and the potential function still lies far ahead. For now, we choose the parameters for the potential function to represent our general knowledge about the world in an ad hoc way based on the following observation: The higher the energy measure of a configuration, the less likely it is to occur. We assign high potential values to those "improbable" subconfigurations over cliques and low potentials to those subconfigurations that we expect to see frequently. Our experimental work is showing satisfactory results (Section 5).

3. Combining Opinions of Early Visual Modules

Most research on evidence combination has focused on updating the "belief" in a given hypothesis about an individual element when a piece of new evidence becomes available [Pearl, 86] [Shafer, 76] [Reynolds *et al* , 86]. This approach, however, is not suitable for our purpose. First of all, in our model the prior knowledge about the image elements is global and statistical - it is their joint probability distribution. In contrast, a piece of evidence from local observation bears directly upon an individual element. It is not computationally feasible to maintain "marginal belief" for individual image elements by summing the joint probabilities whenever a new piece of evidence becomes available. Moreover, a piece of new evidence about a particular element influences the belief distributions of the rest of elements. We believe that an information fusion mechanism should constantly maintain a representation of knowledge to reflect the total information available, except possibly transient periods of time for aggregating evidence locally. This requirement is also desirable, if not necessary, for implementing vision systems in distributed environments. Maintaining "marginal belief" requires the effects of updating local "belief" to be spatially propagated, thus violating such a requirement.

In this section, we limit our attention to an individual element s of S. We show how the opinions about s can be combined and provide the probabilistic justification for the proposed method. In Section 4 we show how the updating of this joint probability distribution given a new set of opinions about a set of primitive elements can be carried out with simple operations.

3.1. Representations and Combination Rules

As in Pearl's construction [Pearl, 86], we assume the segmentation labels can be organized as a hierarchical tree H (e.g. Figure 1). Node l denotes the hypothesis that the corresponding primitive element is of label l, i.e., $X_s = l$. Each internal node stands for the disjunction of its sons. Unlike other approaches, the numbers maintained in our method do not indicate the states of belief of the hypotheses but rather the degrees of hypothesis confirmation or disconfirmation provided by the collected evidence. In particular we do not require an initial "belief" (prior probability) for each node as required in Pearl's scheme.

Let α_l denote the current degree of confirmation/disconfirmation for node l. The probabilistic interpretations for the α's will be given in Section 3.2. Initially, α_l is set to unity for every l indicating "neither confirmed nor disconfirmed". Besides α, each internal node l keeps one value, w_i^l, for each son i. Initially, w_i^l is set to the *a prior* probability of i given l. Obviously, the w_i^l's of each node sum up to unity initially.

Suppose a module A reports its opinion as a set of likelihood ratios $\{\lambda_l^A | l \in L_A\}$ where L_A is a set of mutually exclusive labels contained in H as described in Section 1.2. The corresponding α's are updated according to the following rules:

$$\alpha_l \leftarrow \lambda_l^A \alpha_l \quad \text{for each } l \in L_A \qquad (3.1)$$

To maintain the coherence of the α's, the effect of this opinion has to be propagated throughout the label tree. The propagation procedure can be described in terms of message passing:

(1) Every node l, $l \in L_A$, sends a message, $m = \lambda_l^A$, to its father and each of its sons.

(2) Any node k that receives a message m from its father, passes m to all its sons and replaces α_k by $m\alpha_k$, that is,

$$\alpha_k \leftarrow \lambda_k^A \alpha_k \tag{3.2}$$

(3) Any node j that receives a message m from one of its sons (say i), updates w_i^j by mw_i^j, i.e.,

$$w_i^j \leftarrow mw_i^j \tag{3.3}$$

and sends a message m' to its father, where

$$m' = \sum_i w_i^j \tag{3.4}$$

then updates α_j and all the w^j's according to

$$\alpha_j \leftarrow m'\alpha_j \tag{3.5a}$$

$$w_k^j \leftarrow \frac{w_k^j}{m'} \text{ for all } k \tag{3.5b}$$

where the summation in (3.4) is taken over all the sons of j.

It is easy to see that the combination and propagation procedures are commutative and associative, so their order is irrelevant.

3.2. Probabilistic Justification

The above method fits in the Bayesian formalism if we maintain two notions of conditional independence. First, evidence O^A that bears directly on a label l says nothing about the descendants of l:

$$P(O^A|l,l_i) = P(O^A|l), \quad l_i \text{ descendant of } l, \tag{3.6}$$

$$P(O^A|\neg l,l_j) = P(O^A|\neg l) \quad l_j \text{ descendant of } \neg l$$

Second, the observations of different modules are conditionally independent.

$$\frac{P(O|l)}{P(O|\neg l)} = \prod_A \frac{P(O^A|l)}{P(O^A|\neg l)}, \tag{3.7}$$

where the product on the right-hand side is over a set of modules and O is the union of their observation O^A's.

As suggested by Pearl, (3.6) states that when the observation O^A is a unique property of l, common to all its descendants, once we know l is true/false, the identity of l_i or l_j does not make O^A more or less likely. (3.7), implicitly used in Pearl's scheme, states that each piece of evidence observed by the early modules provides independent information about a label. We believe that the disparate types of image clues in vision applications satisfy this assumption.

We define *consistent states* of α's as the states in which for each available opinion, all of the α's are either updated according to rules (3.1) - (3.5), or none of the α's have been changed with respect to this opinion. We say that a set of opinions *derives* a consistent state if all opinions in this set, and no other opinions, have been used to update the α's. The following theorem relates the α's to the likelihood probabilities at consistent states.

Theorem 1: Let α_t^l denote the α value for l at the consistent state t, and $P(O_t|l)$ be the probability of O_t given the label l, where O_t denotes the union of those observations that form the set of opinions that derives the state t. If $O_t \neq \emptyset$, then

$$\alpha_t^l = c_t P(O_t|l) \quad \text{for all } l \in H \tag{3.8}$$

where c_t is a constant depending only on t, given (3.6) and (3.7).

Due to the limitation on the size of this paper, we provide here several keys for proving Theorem 1 instead of a complete proof. The effect of applying rule (3.1) is to multiply each label l in L_A by the likelihood $P(O^A|l)$ and labels in $\neg L_A$ by $P(O^A|\neg(\bigcup_{l\in L_A} l))$. Downward propagation, rule (3.2), follows directly from the conditional independence assumptions (3.6). Upward propagation, rule (3.3) - (3.5), follows from the fact:

$$P(O|l) = \sum_{l_i} P(O|l_i)P(l_i|l)$$

where l is the conjunction of l_i's.

Applying Theorem 1 and Bayes' rule, we have:

Corollary 1: Let α_t^l denote the α value for l at the consistent state t. If P_0 is the prior *p.d.f.* of a set of mutually exclusive and exhaustive labels L, then the posterior probability $P_t(l)$ of $l \in L$ at the consistent state t is

$$P_t(l) = \frac{\alpha_t^l P_0(l)}{\sum_{l\in L} \alpha_t^l P_0(l)}.$$

To summarize: We have developed an evidence combination method for a hierarchy of hypotheses based on the notions of conditional independence given by (3.6) and (3.7). This scheme, besides having all the characteristics listed in [Pearl, 86], has the following advantages:

(1) The computations involved are extremely simple. Simpler and fewer messages must be passed. Normalizations are never needed since relative degrees of confirmation/disconfirmation are maintained instead of probabilities (Theorem 1).

(2) This scheme decouples the notion of evidence and *a priori* belief. That is, the evidence can be collected and combined in the absence of *a priori* belief in a probabilistic formalism. In the next section we show this characteristic is very helpful when the prior knowledge is represented as an MRF.

4. Combining Prior Knowledge with Observations

In this section, we move our attention to the relationships of segments of the image S. Recall that in the last section, each primitive element is associated with a set of α's to maintain the opinions of the early visual modules. Let β_s denote the set of α's associated with $s \in S$, and $\beta_s(l)$ be the α value for label l in β_s. Define a *global consistent state* to be a state of the β's at which each β_s is in a consistent state.

Assume that the prior knowledge about the image is represented as an MRF X over S, $X_s \in L$ - a mutually exclusive and exhaustive label set in H, with respect to a neighborhood system N. The Gibbs measure that characterizes the prior MRF is:

$$P_\theta(\omega) = \frac{e^{-\frac{1}{T}U_\theta(\omega)}}{Z_\theta} \tag{4.1}$$

where

$$U_\theta(\omega) = \sum_{c \in C} V_c(\omega)$$

Given (1.1), Theorem 1, and Bayes' rule, the *a posterior* Gibbs measure of a configuration ω at a global consistent state t can be computed as:

$$P_t(\omega) = \frac{e^{-\frac{1}{T}U_t(\omega)}}{Z_t} \tag{4.2}$$

where the *a posteriori* energy is

$$U_t(\omega) = \sum_{c \in C} V_c(\omega) - T \sum_{s \in S} \ln(\beta_s^t(\omega_s)) \tag{4.3}$$

It is easy to see that the *a posteriori* Gibbs measure characterizes a MRF over S with respect the neighborhood system N with the local characteristics:

$$P_t(X_s = \omega_s | X_r = \omega_r, r \in N_s) = \frac{e^{-\frac{1}{T}\sum_{c \in C_s} V_c(\omega) + \ln(\beta_s^t(\omega_s))}}{\sum_{\omega'} e^{-\frac{1}{T}\sum_{c \in C_s} V_c(\omega') + \ln(\beta_s^t(\omega'_s))}} \tag{4.4}$$

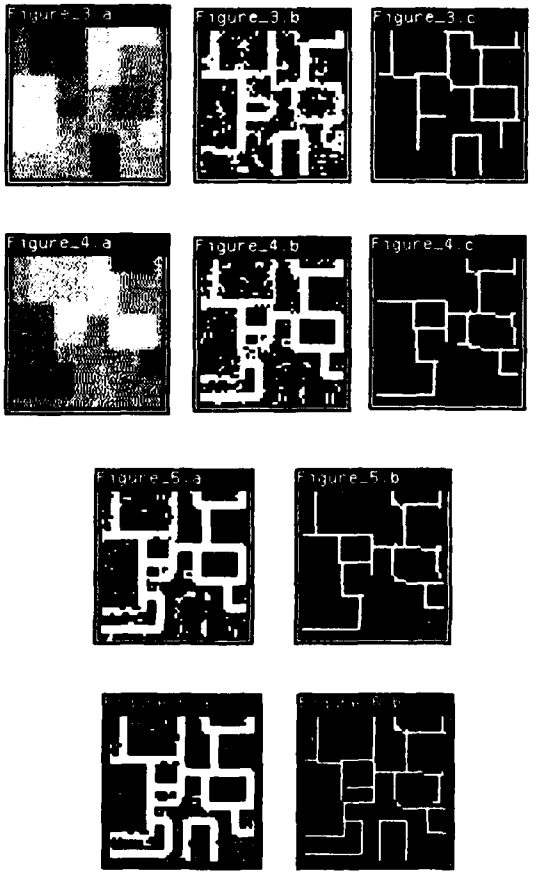
(4.3) and (4.4) indicate that only simple local operations are needed to update the energy measure and local characteristics as new opinions from the early visual modules become available. Therefore, estimation methods depending only upon these measures, such as MAP and MPM estimations, can easily be implemented in the proposed framework (Section 5). We believe that based on this property, novel estimation algorithms can ultimately be designed that incrementally improve their estimations as more and more information arrives. For now, the existing Bayesian estimation methods can be invoked at any global consistent state to provide the up-to-date estimations.

5. Experimental Results

We demonstrate the method using two images of overlapping rectangular patches. Each patch in the first image corresponds to a geometrically identical patch in the second. The intensities of the patches in each image are randomly selected from the range [0, 255], with no intensity correlation between images. A random number is generated for each pixel and is added to its intensity value to simulate a random noise process. The distributions used for the noise in the two images are Gaussian of zero mean, with standard deviation 16 and 12 respectively. These two images can be considered as two different sources of information about the same set of rectangular objects. Figure 3.a and 4.a are halftone displays of these two images.

A set of likelihood edge detectors, an early version of the detectors described in [Sher, 86], provides a set of likelihood ratios - $\{\frac{P(O|E_i)}{P(O|NE)} | i=1,2,3,4\}$ - for each pixel given the 3×3 window of intensities centered at it, where NE denotes the hypothesis that the given pixel is not an edge element, and E_i denotes the hypothesis that the given pixel is an edge of one of the four (horizontal, vertical, and two diagonal) orientations. The likelihoods are computed using a model for step edges and a Gaussian model for additive noise. Figure 3.b and 4.b show the Maximum Likelihood Estimation (MLE) for edges in Figure 3.a and 4.a respectively. That is, a pixel s is on if and only if $\max_l \beta_s(l) > \beta_s(NE)$.

We use a homogeneous and isotropic MRF with a third order neighborhood system over the image lattice to encode a body of basic knowledge about edges. Cliques of size 3 are used to discourage parallel and competing edges, whereas cliques of size 2 are used to encourage line continuations, region homogeneity and to discourage breaks in the line forming process. The potential assignments for the cliques are chosen conservatively in the sense that estimation methods based on (4.2) and (4.3) make as few false detections of edges as possible while maintaining reasonable detectability.



We have implemented a program that does MPM estimation based on the Monte Carlo procedure proposed in [Marroquin, 85]. This program uses the Gibbs sampler that randomly and repeatedly selects configurations for each pixel based on the distribution given in (4.4). Due to the fast convergence observed in the experiments, we start collecting statistics at the 300th iteration. Figure 3.c and 4.c show the MPM estimations based on the statistics collected over 300 iterations.

We can consider that Figures 3.a and 4.a provide only partial evidence to support the NE and E_i's hypotheses. For example, Figure 3.a would represent a noise-corrupted depth map, Figure 4.a would be an image of irradiance, and E_i would correspond to the hypotheses that the pixel is at a depth discontinuity or at an edge purely caused by photometrical effects. Under this interpretation, Figure 5.a and 5.b show the MLE and the MPM estimations resulting from applying the upward propagation rule (3.3)-(3.5) with the initial $w_D = 0.75$, where D stands for depth edge, to combine the partial evidence of Figure 3.a with the information provided by Figure 4.a.

Alternatively we can consider that Figure 3.a and 4.a independently support the same set of hypotheses. By applying rule (3.1), we obtain Figure 6.a and 6.b representing the MLE and the MPM estimations based on the combined information. Observe the lines detected in the lower left quadrant of Figure 6.b that do not show up in either of Figure 3.c and 4.c, and the false detections in Figure 3.c and 4.c that are removed in Figure 6.b. These sorts of results can not be achieved by multi-modal segmenters that rely on Boolean operations to combine evidence.

6. Conclusions and Future Research

We have presented a new approach, based on Bayesian probability theory, for integrating information from various sources. This approach decouples the notion of observable evidence and prior knowledge so that each type of information can be treated according to its own characteristics. We have shown how to combine coherently and consistently different pieces of evidence for a set of hypotheses organized as a hierarchical knowledge tree and how to incorporate prior knowledge modeled by MRFs. We have explicitly listed the set of independence assumptions we need and have provided the probabilistic justification for this approach. Experiments on synthetic data have shown promising results. There is still much to do, however. The MRF model we used is very rudimentary. Much more prior knowledge can be encoded using the same concept. We are currently improving the MRF model to handle curved lines. One of our ultimate goals is to encode all kinds of geometrical and photometrical constraints in terms of local clique potentials in an MRF that has general connectivity. The stochastic estimation methods are computationally very expensive. We are now designing a deterministic estimation algorithm that incrementally improves its estimation as new evidence arrives. We believe this method of information fusion can be applied to problems other than image segmentation as well.

Ackowledgments

We would like to thank Dave Sher for providing his edge detectors and many valuable suggestions. We also thank Rajeev Raman for his help in writing the code for the experiments.

This work was supported by the Air Force Systems Command, Rome Air Development Center, Griffiss Air Force Base, New York 13441-5700, and the Air Force Office of Scientific Research, Bolling AFB, DC 20332 under Contract No. F30602-85-C-0008. This contract supports the Northeast Artificial Intelligence Consortium (NAIC). This work was also supported by the U.S. Army Engineering Topographic Laboratories under Contract No. DACA76-85-C-0001

References:

Barrow, H. G., and J. M. Tenenbaum, Recovering intrinsic scene characteristics from images, in *Computer Vision Systems*, A. R. Hanson and E. M. Riseman ed., Academic Press, 1978.

Bolle, R. M. and D. B. Cooper, Bayesian Recognition of Local 3-D Shape by Approximating Image Intensity Functions with Quadric Polynomials, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-6, No. 4, pp. 418-429, July 1984.

Bolles, Robert C.: Verification Vision for Programmable Assembly, *Proceedings*: IJCAI, 1977.

Cross, G. R., and A. K. Jain, Markov Random Field Texture Models, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, PAMI-5, 25-39, 1983.

Derin, H., and W. S. Cole, Segmentation of Textured Images Using Gibbs Random Fields, *Computer Vision, Graphics, and Image Processing* 35, 72-98, 1986.

Elliott, H., and H. Derin, Modelling and Segmentation of Noisy and Textured Images Using Gibbs Random Fields, Technical Report #ECE-UMASS-SE84-1, University of Massachusetts, 1984.

Feldman, J. A., and Y. Yakimovsky, Decision Theory and Artificial Intelligence: I. A Semantics-Based Region Analyzer, *Artificial Intelligence* 5, 349-371, 1974.

Geman, Stuart and Donald Geman, Stochastic Relaxation, Gibbs Distributions, and the Baysian Restoration of Images, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, PAMI-6, No. 6, 1984.

Grimson, W. E. L., From image to Surfaces: A computational study of the human early visual system, Cambrige, MA, MIT Press, 1981.

Hassner, Martin and Jack Slansky, The Use of Markov Random Fields as Models of Texture, in *Image Modeling*, Azriel Rosenfeld (ed.), Academic Press, Inc., 1980, 185-198.

Hummel, R., and S. Zucker, On the Foundation of Ralaxation Labeling Process, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-5, 1983, 267-286.

Ikeuchi, K., and B. K. P. Horn, Numerical shape from shading and occluding boundaries. *Artificial Intelligence*, 15, 141-184, 1981.

Kindermann, Ross and J. Laurie Snell, Markov Random Fields and their Applications, American Mathematical Society, 1980.

Marr, David, *Vision*, San Francisco: W. H. Freeman, 1982.

Marroquin, Jose L., Probabilistic Solution of Inverse Problems, Technical Report 860, MIT Artificial Intelligence Laboratory, 1985.

Pearl, Judea: On Evidential Reasoning in a Hierarchy of Hypotheses, *Artificial Intelligence*, **16**, No.2, Feb. 1986.

Rosenfeld, A., R. A. Hummel, and S. W. Zucker, Scene labelling by relaxation operations. *IEEE Trans. SMC* **6**, 1976, 420-433.

Shafer, G., *A Mathematical Theory of Evidence*, Princeton University Press, 1976.

Sher, David, Optimal Likelihood Detectors for Boundary Detection Under Gaussian Additive Noise, *Proceedings*: IEEE Conference on Computer Vision and Pattern Recognition, Miami, Florida, June 1986.

Sher, David, Advanced Likelihood Generators for Boundary Detection, Technical Report 197, Computer Science Department, University of Rochester, 1987.

Shvaytser, H., and Shmuel Peleg, A New Approach to the Consistent Labeling Problem, *Proceedings*: IEEE Conference on Computer Vision and Pattern Recognition, 85, 320-327.

Terzopoulos, D., Integrating Visual Information from Multiple Sources, in *From Pixel to Predicates*, Alex P. Pentland (ed.), Ablex Publishing Co., 1986, 111-142.

Witkin, Andrew, Recovering Surface Shape and Orientation from Texture, *Artificial Intelligence* **17**, 1981, 17-45.

MINIMIZATION OF THE QUANTIZATION ERROR IN CAMERA CALIBRATION

Behrooz Kamgar-Parsi

Center for Automation Research
University of Maryland
College Park, MD 20742

ABSTRACT

It is shown that the quantization error due to the limited resolution capabilities of existing cameras is so serious that the calibration of two camera stereo in an uncontrolled environment is frequently unreliable. This is done by deriving explicit analytical solutions for the relative pan and tilt angles in terms of the world pan angle (often referred to as gaze angle) and the coordinates of the image points used in their computation (under the assumption that an estimate of these angles is available). The degree of the impact of the quantization error, however, depends highly on the image points that are used for the computation of the camera orientations. That is, to obtain reliable results the image points must be discriminated on the basis of their coordinates and, at times, their horizontal disparities. Not only this, but also the combination of image points used for the computation of a given angle must form special configurations. These findings, obtained by deriving analytical expressions for the magnitude of the error in the computed values of the orientation angles, indicate that points and combinations of points used for calibration must be selected intelligently. A scheme is presented to choose (from the available image points) the appropriate points and combinations of points that give rise to the most accurate solution.

1. INTRODUCTION

Tasks such as scene analysis, eye-hand coordination and navigation require depth information. A standard way of obtaining depth information is to extract it from stereo image pairs. For example, this technique can be used on an autonomous vehicle which would "look" at the road using two cameras and create a stereo pair of images. Depth information would then be used to build a 3-D representation of the world facing the vehicle. Utilizing this 3-D representation, the vehicle would be able to plan ahead and follow the road.

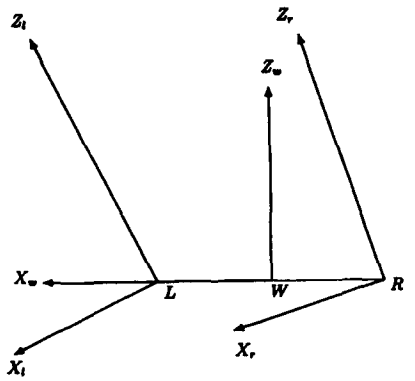
The accuracy of the 3-D representation, however, depends on the accuracy with which the orientations of the two cameras are known. Although for nearby objects small inaccuracies in camera orientations may not be of much significance, for objects that are not close to the stereo system the situation is quite different. In fact, in the presence of inaccuracies in camera orientations, the accuracy of the 3-D representation can deteriorate drastically with depth. This, in turn, will curtail the capabilities of the vehicle to plan ahead or to plan correctly.

Some investigators have addressed the topic of stereo error due to quantization but they have not discussed its impact on the problem of camera calibration [1], [2]. Other investigators have discussed the question of camera calibration without considering the effect of quantization error thereby limiting the practicality of their approach [3], [4]. In addition to stereo error analysis, we have discussed practical difficulties in camera calibration and ways to handle them.

It is shown [5] that, in stereo, precise knowledge of the relative pan angle of the two cameras with respect to each other (and to a lesser degree the relative tilt angle) is crucial to the accurate 3-D recovery of object points in space, whereas accurate knowledge of the pan and tilt angles relative to the scene, i.e. the "world" angles, is of less significance. Therefore, the most important task would be the computation of the relative pan and tilt angles. If possible the world pan angle should also be computed. The world tilt angle cannot be computed.

It is assumed that the stereo system consists of two identical and synchronized cameras that can pan and tilt independently, but are installed at a fixed distance from one another. The fixed line segment defined by the centers of rotation of the two cameras is called the baseline, and the world coordinate system is defined as follows: Take the center of the coordinate system to be the point midway between the centers of rotation of the two cameras, the X-axis to point from the right camera to the left camera, the Y-axis to be vertically upward and the Z-axis straight ahead. Similar axis orientations are observed for coordinate systems attached to the right and the left cameras. The center of the coordinate system for either camera will be its center of rotation.

Let θ be the pan (counterclockwise Y rotation) and ϕ the tilt (counterclockwise X rotation) of the left camera coordinate system with respect to the right camera, and let α be the counterclockwise pan angle of the right camera



This diagram shows the orientation of the X and Z axes for the world, the right and the left coordinate systems, whose centers are points W, R, and L, respectively. The angle between X_w and X_r is α, and the angle between X_w and X_l is $\alpha + \theta$.

Figure 1

with respect to the world coordinate system.

If (X_r, Y_r, Z_r) are the coordinates of a given point in space with respect to the right camera coordinate system and (X_l, Y_l, Z_l) are those with respect to the left coordinate system, we have

$$\begin{pmatrix} X_r \\ Y_r \\ Z_r \end{pmatrix} = \begin{pmatrix} \cos\theta & \sin\theta\sin\phi & \sin\theta\cos\phi \\ 0 & \cos\phi & -\sin\phi \\ -\sin\theta & \cos\theta\sin\phi & \cos\theta\cos\phi \end{pmatrix} \begin{pmatrix} X_l \\ Y_l \\ Z_l \end{pmatrix} + \begin{pmatrix} b\cos\alpha \\ 0 \\ b\sin\alpha \end{pmatrix} \quad (1)$$

where b is the length of the baseline.

It is obvious that there is no advantage to allowing the difference in the tilt angles of the two cameras, i.e. ϕ, to be large. Also, we assume that an estimate, θ_0 of θ is available. That is, we have $\theta = \theta_0 + d\theta$, where $d\theta$ is a small angle.

It is shown that precise knowledge of θ (and to some degree ϕ) is crucial to the accurate 3-D recovery of object points in space, whereas accurate knowledge of α is of less importance. On the other hand, as we shall see, compared to θ and ϕ, the numerical computation of α is more likely to produce an erroneous result. Therefore, although the objective is to compute all three angles, we have shown that it is often possible to bypass the calculation of α and to compute θ and ϕ directly. This is despite the fact that in the analytic formulation of the problem the three angles are intertwined. It has to be noted that, in general, we have assumed that an estimate of the angle α is available.

2. ANALYTICAL EXPRESSIONS FOR RELATIVE PAN AND TILT ANGLES

After expanding the sine and the cosine functions (aside from those for α) in equation (1) and ignoring terms in the second order in $d\theta$ and ϕ, that equation may be written as

$$x_r\lambda_r = (x_l\cos\theta_0 - f\sin\theta_0)\lambda_l - (f\cos\theta_0 + x_l\sin\theta_0)\lambda_l d\theta + b\cos\alpha \quad (2)$$

$$y_r\lambda_r = y_l\lambda_l + f\lambda_l\phi \quad (3)$$

$$-f\lambda_r = -(x_l\cos\theta_0 - f\sin\theta_0)\lambda_l d\theta + y_l\lambda_l\phi - (f\cos\theta_0 + x_l\sin\theta_0)\lambda_l + b\sin\alpha \quad (4)$$

where f is the focal length of either camera, and (x_r, y_r) and (x_l, y_l) are the right and the left projected coordinates of a given point in space. That is, $x_r = -fX_r/Z_r$ and $y_r = -fY_r/Z_r$. x_l and y_l are defined similarly. λ_r and λ_l are defined as $-Z_r/f$ and $-Z_l/f$, respectively.

Assuming, for the moment, that α is known, we have three equations in four unknowns, i.e. $d\theta, \phi, \lambda_r$ and λ_l. We can rewrite equations (2) and (3) as follows:

$$d\theta = \frac{x_l\lambda_l - x_r\lambda_r + y_l\lambda_l\sin\theta_0\phi + b\cos\alpha}{(f\cos\theta_0 + x_l\sin\theta_0)\lambda_l} \quad (5)$$

$$\phi = \frac{y_r\lambda_r - y_l\lambda_l}{f\lambda_l} \quad (6)$$

Substitution of (5) and (6) into (4) will yield an expression for λ_r in terms of λ_l. This can in turn be substituted into (5) and (6) to provide expressions for $d\theta$ and ϕ in terms of λ_l only. Doing this we will have two equations in three unknowns, i.e. $d\theta$, ϕ and λ_l. Considering a second point in space (with known right and left image positions), we will have altogether four equations in the following four unknowns: $d\theta$, ϕ, λ_{1l} and λ_{2l}. After considerable algebra λ_{1l} and λ_{2l} are eliminated and the expressions for $d\theta$ and ϕ simplify to these:

$$d\theta = \frac{(1 - P_{12})(a'_{2l}y_{2r} - a_{2r}y_{2l})[(f^2 + y_{1r}y_{1l}\cos\theta_0)\cos\alpha + (fx_{1r} - y_{1r}y_{1l}\sin\theta_0)\sin\alpha]}{y_{1r}b'_{1l}[fa_{2r} + y_{2r}y_{2l}\cos(\alpha + \theta_0)] - y_{2r}b'_{2l}[fa_{1r} + y_{1r}y_{1l}\cos(\alpha + \theta_0)]} \quad (7)$$

$$\phi = \frac{y_{1l}y_{2r}a_{1r}b_{2l} - y_{2l}y_{1r}a_{2r}b_{1l} + y_{1r}y_{2r}(c_{2l}d_{1l} - c_{1l}d_{2l})}{y_{1r}b'_{1l}[fa_{2r} + y_{2r}y_{2l}\cos(\alpha + \theta_0)] - y_{2r}b'_{2l}[fa_{1r} + y_{1r}y_{1l}\cos(\alpha + \theta_0)]} \quad (8)$$

where P_{12} is a permutation operator (switching the indices 1.and 2 in the expression to which it is applied), and where

$$a_{ir} = f\cos\alpha + x_{ir}\sin\alpha \quad (9)$$

$$a_{il} = f\cos\alpha + x_{il}\sin\alpha \quad (10)$$

$$b_{il} = x_{il}\cos\alpha - f\sin\alpha \quad (11)$$

$$c_{il} = f\cos\theta_0 + x_{il}\sin\theta_0 \quad (12)$$

$$d_{il} = x_{il}\cos\theta_0 - f\sin\theta_0 \quad (13)$$

$$a'_{il} = c_{il}\cos\alpha + d_{il}\sin\alpha \quad (14)$$

$$b'_{il} = d_{il}\cos\alpha - c_{il}\sin\alpha \quad (15)$$

Also the following definitions will be useful:

$$\gamma_i = x_{ir}y_{il} - x_{il}y_{ir} \quad (16)$$

$$\delta_{iy} = y_{ir} - y_{il} \quad (17)$$

$$p_i = f^2 + y_{ir}y_{il} \quad (18)$$

$$\gamma'_i = x_{ir}y_{il} - d_{il}y_{ir} \quad (19)$$

$$p'_i = f^2 + y_{ir}y_{il}\cos\theta_0 \quad (20)$$

Numerical computation of $d\theta$ and ϕ is composed of the calculation of several intermediate quantities (such as γ, δ, etc.). In order to discuss the computational difficulties, we

need to examine the magnitudes of these quantities separately. Presently these quantities are expressed in terms of x_r, x_l, y_r and y_l (of a given point in space), which makes it difficult to estimate their magnitudes. This is because for a point in space not all x and y coordinates of both left and right image positions can be considered as independent quantities. For our stereo system, in particular, y_r and y_l are tightly correlated. Therefore in Appendix A we have eliminated y_l and have derived expressions for the intermediate quantities in terms of $d\theta, \phi, x_r, x_l$ and y_r.

3. COMPUTATION OF RELATIVE PAN ANGLE WHEN WORLD PAN ANGLE IS ZERO

To emphasize the computational difficulties, first we focus on the computation of $d\theta$ for the less complicated case of $\theta_0 = \alpha = 0$. Necessary modifications for the case of arbitrary values of θ_0 and α will be discussed later.

Setting α to zero, equation (7) reduces to

$$d\theta = \frac{f[(y_{2r} - y_{2l})(f^2 + y_{1r}y_{1l}) - (y_{1r} - y_{1l})(f^2 + y_{2r}y_{2l})]}{x_{1l}y_{1r}(f^2 + y_{2r}y_{2l}) - x_{2l}y_{2r}(f^2 + y_{1r}y_{1l})} \tag{21}$$

If we were dealing with a system where x's and y's could be obtained with high precision, then, using the above expression, any two points in space (which did not happen to give rise to a vanishing denominator) could give us an accurate value for $d\theta$.

However, x's and y's can, at best, be obtained to the precision of a pixel. This makes the numerical computation of the angle $d\theta$ a difficult task. Precautions have to be taken to ensure that all the mathematical operations required during the computation of $d\theta$ produce acceptable values. The only operations involved are additions, subtractions, multiplications and divisions. As is the case in most numerical computations involving these four operators, the two that are most likely to intensify errors are subtractions and divisions. Divisions are prone to producing unacceptable results when their denominators are small. Subtractions, however, may lead to inaccurate results when their operands are almost equal.

Computation of $d\theta$ requires one division and several subtractions. To know whether the computation will be reliable or not, one needs to examine the magnitudes of the operands of each of these operators. Among the subtractions, two are those that define vertical disparities of the points in space. We must therefore find out how reliable the computation of vertical disparity is. Assuming that correct matches have been found, y_r and y_l can still be erroneous up to half a pixel. This implies that vertical disparity can be erroneous up to a full pixel. To know how significant this error is we need to have an estimate of the magnitude of the vertical disparity. For this, in equation (A7), we set α to zero. To the first order in $d\theta$ and ϕ that equation may be written as

$$y_r - y_l \simeq (f^2 + y_l^2)\phi/f - x_l y_l d\theta/f \tag{22}$$

Note that if we start from equation (A6), then instead of y_l on the right hand side of equation (22) we will have y_r. The rest of the expression will be the same. Now, if ϕ is negligible, then

$$\delta_y \simeq -(x_l y_l/f)d\theta \tag{23}$$

To see how small this quantity is we write θ in terms of degrees and f in terms of n and v, where n is the number of pixels on every row (or column) of the image plane and v the view angle of the camera:

$$\delta_y \simeq -\frac{x_l y_l}{(n/2)\cot(v/2)}(.017\theta) \tag{24}$$

y_r, y_l and x_l are now in (units of) pixels.

For a typical camera we have roughly $n = 512$ and v=50 degrees. For the largest possible value that x_l and y_r can have, i.e. 256 pixels, the vertical disparity will be $|y_r - y_l| = 2.08\theta$. For θ=.5 degrees, the vertical disparity will, therefore, be 1 pixel and that is within the range of error. This may suggest to increase the accuracy of the computation of θ, one should attempt to enhance the magnitude of the vertical disparity by increasing the vergence angle or by keeping the other term in equation (22). The latter is possible by intentionally keeping the two cameras at different tilts. Such strategies, however, are not significantly helpful. This is because in the numerator of (21) the two expressions $f^2 + y_{1r}y_{1l}$ and $f^2 + y_{2r}y_{2l}$ are roughly of equal magnitudes. Therefore, the middle subtraction in the numerator of equation (21) is basically a subtraction of two vertical disparities implying that any attempt to increase vertical disparities would leave the difference between the two terms $(y_{2r} - y_{2l})(f^2 + y_{1r}y_{1l})$ and $(y_{1r} - y_{1l})(f^2 + y_{2r}y_{2l})$ relatively unchanged.

We now discuss the effect of the other subtractions. In the numerator of equation (21), aside from the two subtractions that define vertical disparities of the two points in space, there is only one more subtraction. It should be noted, however, that there is a hidden subtraction which does not appear in (21) and yet requires attention. Rearranging terms in the numerator of equation (21), we can write

$$d\theta = \frac{f[(y_{1l} - y_{2l})(f^2 + y_{1r}y_{2r}) - (y_{1r} - y_{2r})(f^2 + y_{1l}y_{2l})]}{x_{1l}y_{1r}(f^2 + y_{2r}y_{2l}) - x_{2l}y_{2r}(f^2 + y_{1r}y_{1l})} \tag{25}$$

Here we note that a successful computation of $d\theta$ requires large values of $y_{2r} - y_{1r}$ (and also $y_{2l} - y_{1l}$).

This issue is easy to resolve: Points that are paired for computation of $d\theta$ should be vertically far apart. This implies that we must not, in an attempt to improve our estimation of $d\theta$, pair every point with known right and left image positions with all others (and then average). It is much better to average over a relatively few well paired points.

We now go back to equation (21) and examine the remaining subtraction in the numerator, i.e. the middle one. Unfortunately, the operands of this subtraction are almost equal causing the subtraction to easily produce erroneous results. To minimize the adverse effects of this subtraction, we need to make the two operands as different as we can by choosing appropriate points and pairs of points. To find out where these points are located, we use equation (A10) which does not involve y_l. After setting $\alpha = 0$, we find the analytical result of the subtraction in question, num (which is essentially the numerator in the expression for $d\theta$), to be the following:

$$num = [x_{1l}y_{1r}(f^2 + y_{2r}^2) - x_{2l}y_{2r}(f^2 + y_{1r}^2)]d\theta/f \qquad (26)$$

Ideally we want to maximize num by choosing appropriate values for the four quantities, x_{1l}, x_{2l}, y_{1r} and y_{2r}. The only constraint is that y_{1r} and y_{2r} must be considerably different. With this in mind one can easily verify that num can be maximized by setting $x_{1l} = x_{2l} = a$, and $y_{1r} = -y_{2r} = a$, where $2a$ is the length (or the width) of the image plane. Practically speaking this means that on the left image we should look for image points near the four corners (this is because y_l is not too different from y_r), and calculate vertical disparity by finding their matches on the right image. We should then pair points near the top right corner of the left image with points near the bottom right corner of that image. Likewise, we pair image points near the top left corner with points near the bottom left corner. It is easy to verify that so long as we choose such pairs of points the denominator in the expression for $d\theta$ cannot be small, implying that the division involved in the computation of $d\theta$ will be basically an error-free operation.

Now that we have decided what image points we want to use, it is possible to make a rough approximation of the error we would incur in the computation of θ. This is done in Appendix B. The result is that on the average while using the prescribed points the error is .12 degrees. To know how serious the quantization error can be if points close to the middle of the image are used, we note that one can derive the following expression for the error Δ_θ in θ:

$$\Delta_\theta = \frac{f(\Delta_{2y} - \Delta_{1y})}{2a^2}$$

where Δ_{2y} and Δ_{2y} are the errors in the vertical disparities of points 1 and 2.

From the above expression the maximum error (for corner points) is found to be $f/(a^2) \simeq .49$ degrees. Now if we use the same point configuration but pick the points closer to the center of the image, then a in the above formulas will be smaller indicating that the quantization error is larger. For example, if we choose the points that are located half way between the corners and the center of the image, then a will be half what it is for the corner points, and both the average error and the maximum error will four times larger.

It should be noted that in the absence of image points from corners with the same latitude, image points from corners having the same altitude may be paired provided that their y's are sufficiently different.

3.1. Modifications for Large Values of Vergence Angle

If we set $\alpha = 0$, equation (7) reduces to

$$d\theta = \frac{(c_{2l}y_{2r} - fy_{2l})(f^2 + y_{1r}y_{1l}\cos\theta_0) - (c_{1l}y_{1r} - fy_{1l})(f^2 + y_{2r}y_{2l}\cos\theta_0)}{d_{1l}y_{1r}(f^2 + y_{2r}y_{2l}\cos\theta_0) - d_{2l}y_{2r}(f^2 + y_{1r}y_{1l}\cos\theta_0)} \qquad (27)$$

In analogy with the case $\theta_0 = \alpha = 0$, we conclude that the best pairs of points (for the case when θ_0 is not zero) are those for which $y_{1l} \simeq -y_{2l} \simeq a$ and $d_{1l} \simeq d_{2l}$ has its maximum magnitude. Since $d_l = x_l\cos\theta_0 - f\sin\theta_0$, when $\theta_0 > 0$ the two corners that yield $x_l = -a$ are the best neighborhoods from which to choose the two points. Although for small values of θ_0 the other two corners (yielding $x_l = a$) are almost as good, for large values of θ_0 these corners should be avoided. This is because the value of d_l can be too small in these corners. For example, if $\tan\theta_0 \simeq a/f$ then for these corners we have $d_l \simeq 0$. In these situations points that are horizontally near the middle of the image ($x_l \simeq 0$) are preferred to corners with $x_l \simeq a$.

Employing the mathematical tools developed by Kamgar-Parsi and Kamgar-Parsi [6] one can show when points from the appropriate corners are used, then the average value of the digitization error in θ is given by

$$\Delta_\theta = \frac{2f^2k^2 + k^4 + a^2f^2\sin^2\theta_0}{12af^2k(a\cos\theta_0 + f\sin\theta_0)} - \frac{2[(k^2 + af\sin\theta_0)^7 + (k^2 - af\sin\theta_0)^7 - 2k^{14}]}{7!a^5f^5k^6(a\cos\theta_0 + f\sin\theta_0)\sin^2\theta_0}$$

where $k = f\cos\theta_0 - a\sin\theta_0$.

The above expression is good up to $\theta_0 \simeq 30$ degrees. If $\theta_0 \simeq 30$ and the points that are used come from the bad corners the quantization error will be very large. This is because for these corners a (in the above formula) must be replaced by $-a$, causing the expression $a\cos\theta_0 + f\sin\theta_0$ in the denominator to be small. Finally, one can verify that for the case $\theta_0 = 0$ the error given in the above formula reduces to the one derived in Appendix B.

3.2. Modifications for Large Values of World Pan Angle

When α is small the situation is more or less as it is when α is zero. However, for large values of α, terms involving $\sin\alpha$ become significant, and, therefore, their impact on the calculation of the intermediate quantities such as num has to be investigated. From equation (A10) we see that the dependence of num on α is through the a_{ir}'s, a_{il}'s and b_{il}'s. However insensitive the a_{ir}'s and a_{il}'s may be to the value of α, the b_{il}'s, on the other hand, are quite sensitive.

Assuming $\alpha > 0$, for a given negative value of x_{il}, the magnitude of b_{il} will increase as α increases, whereas for a given positive value of x_{il}, the magnitude of b_{il} declines (it can even vanish and then increase again) as α increases. The situation is reversed for $\alpha < 0$.

In other words, standing behind the cameras, if we pan

them to the left (right), then a pair of points from the right (left) corner of the left image will give a more accurate value for $d\theta$ (as compared to $\alpha = 0$), while those from the left (right) corner will give a less accurate value. If $f \sin\alpha$ and $x_{il} \cos\alpha$ are roughly of the same magnitude but of opposite signs, then points that horizontally are closer to the middle of the image plane should be chosen instead.

4. CALCULATION OF WORLD PAN ANGLE

When the world pan angle, α, is not known, then there will be three unknowns, i.e. $d\theta$, ϕ and α. To calculate α we write equation (7) in the following form:

$$d\theta = \frac{A\cos(2\alpha) + B\sin(2\alpha) + C}{D\cos(2\alpha) + E\sin(2\alpha) + F} \tag{28}$$

where

$$A = (1-P_{12})[p_1'(c_{2l}y_{2r} - fy_{2l}) - \gamma_1'(fx_{2r} - y_{2r}y_{2l}\sin\theta_0)] \tag{29}$$

$$B = (1-P_{12})[\gamma_1'p_2' + (c_{2l}y_{2r} - fy_{2l})(fx_{1r} - y_{1r}y_{1l}\sin\theta_0)] \tag{30}$$

$$C = (1-P_{12})[p_1'(c_{2l}y_{2r} - fy_{2l}) + \gamma_1'(fx_{2r} - y_{2r}y_{2l}\sin\theta_0)] \tag{31}$$

$$D = (1-P_{12})[y_{1r}d_{1l}p_2' + y_{1r}c_{1l}(fx_{2r} - y_{2r}y_{2l}\sin\theta_0)] \tag{32}$$

$$E = (1-P_{12})[y_{1r}d_{1l}(fx_{2r} - y_{2r}y_{2l}\sin\theta_0) - y_{1r}c_{1l}p_2'] \tag{33}$$

$$F = (1-P_{12})[y_{1r}d_{1l}p_2' - y_{1r}c_{1l}(fx_{2r} - y_{2r}y_{2l}\sin\theta_0)] \tag{34}$$

In principle, knowing the right and the left image positions of three points in space will allow us to compute two distinct sets of six coefficients in (22), i.e. $A_1, \ldots, F_1$, and $A_2, \ldots, F_2$. The following equation can then be solved for α:

$$\frac{A_1\cos(2\alpha) + B_1\sin(2\alpha) + C_1}{D_1\cos(2\alpha) + E_1\sin(2\alpha) + F_1} = \frac{A_2\cos(2\alpha) + B_2\sin(2\alpha) + C_2}{D_2\cos(2\alpha) + E_2\sin(2\alpha) + F_2} \tag{35}$$

This approach, however, is not too practical. This is because the equation we are dealing with is not linear and its solution may not be stable. Since an estimate, α_0, of the angle α is generally available, we may assume

$$\alpha = \alpha_0 + d\alpha \tag{36}$$

where $d\alpha$ is a small angle. Therefore, we set our task to be the calculation of $d\alpha$, and not α. Since $d\alpha$ is a small angle equation (22) may be written as

$$d\theta = \frac{k + ld\alpha}{m + nd\alpha} \tag{37}$$

where

$$k = A\cos 2\alpha_0 + B\sin 2\alpha_0 + C \tag{38}$$

$$l = 2(B\cos 2\alpha_0 - A\sin 2\alpha_0)d\alpha \tag{39}$$

$$m = D\cos 2\alpha_0 + E\sin 2\alpha_0 + F \tag{40}$$

$$n = 2(E\cos 2\alpha_0 - D\sin 2\alpha_0)d\alpha \tag{41}$$

To calculate each set of k, l, m, n we need two points. Two such sets will then yield

$$d\alpha = \frac{k_2m_1 - k_1m_2}{(k_1n_2 - k_2n_1) + (l_1m_2 - l_2m_1)} \tag{42}$$

Numerical computation of $d\alpha$ is highly sensitive to round-off errors caused by the limited resolution capabilities of the camera. Indeed, for reasons which will become clear shortly, reliable computation of α is a more difficult task than that of θ. To find clues to the successful computation of α, we shall briefly review our approach. We calculate α (or $d\alpha$) such that the two values of $d\theta$ come out equal (see equation (35)). That is, once α is computed, not only must equation (35) hold, but also each side of it has to be equal to the true value of $d\theta$. It is therefore necessary to choose those image points that, except for the error in α, would produce reliable values for θ. This must be observed even if these points are not used for the calculation of θ. Hence, points and pairs of points that are not qualified for the computation of θ shall not qualify for the computation of α either.

We still need to make one further consideration. Unless the two sets of six coefficients involved in equation (35) are sufficiently different, the computation of $d\alpha$ is likely to be overwhelmed by errors. Theoretically, we either need three points that when paired two by two give sets of coefficients that are sufficiently different, or two distinct pairs of points yielding adequately different sets of coefficients. Practically, however, except for unusual situations, the former is not possible (because the three points must still constitute two pairs of points that are appropriate for computation of θ). To avoid confusion, from now on, unless stated otherwise, we assume that the number of points required for a single computation of $d\alpha$ is four.

To know where these four points can be found, we need to examine the six coefficients, A, B, C, D, E, F, and to estimate their magnitudes. To simplify this task here we concentrate on the case $\theta_0 = 0$. Yet, the dependence of these coefficients on a large number of parameters makes it virtually impossible to extract any useful information from the formulas defining them. Nevertheless, we can simplify our task by noting that any set of six coefficients that is appropriate for computation of $d\alpha$ has to be formed by pairing two points that are appropriate for computation of θ, i.e. two points whose projections on the left image plane lie near two corners that have the same latitude. Therefore, for these two points we can roughly assume that $x_{1l} \simeq x_{2l}$, $y_{1l} \simeq -y_{2l}$. Since vertical disparity is not large we can also assume that $y_{1r} \simeq -y_{2r}$.

Furthermore, we will focus our attention only on the coefficients in the denominator, i.e. D, E, and F. We will not examine A, B, and C for two reasons, a) they are still complicated, and b) in general if the D's, E's and F's are sufficiently different so will be the A's, B's and C's (this can be seen from equation (35), and also from the discussion in Appendix C). After replacing x_{1l} and x_{2l} by x_l, y_{1r} and $-y_{2r}$ by y_r, and p_1 and p_2 by p, for the case $\theta_0 = 0$ we may write

$$D \simeq y_r[2px_l + f^2(x_{1r} + x_{2r})] \quad (43)$$

$$E \simeq fy_r[-2p + x_l(x_{1r} + x_{2r})] \quad (44)$$

$$F \simeq y_r[2px_l - f^2(x_{1r} + x_{2r})] \quad (45)$$

From the above equations we can see that by choosing x_{1r} and x_{2r} appropriately, one can obtain two sets of D, E and F that are sufficiently different. We will distinguish two cases depending on whether x_l is close to $-a$ or a (where $2a$ is the length or the width of the image plane).

Case I ($x_l \simeq -a$):

In this case the point in space has to be wide and deep. For, otherwise, it would be outside the view of the right camera. Therefore, the right image will also be close to the corner, implying that x_r will be close to $-a$ too (points 1 and 2 in Figure 2). Therefore, the three coefficients will have the following values:

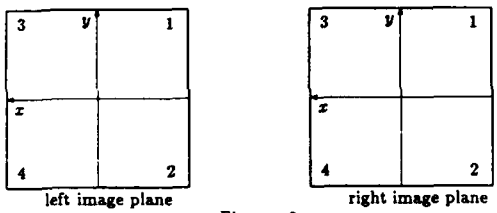


Figure 2

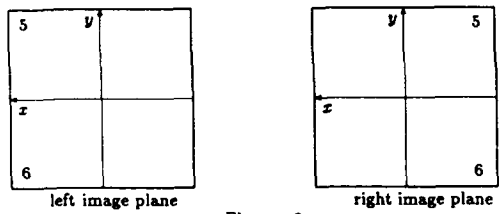


Figure 3

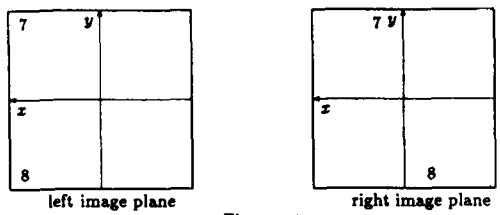


Figure 4

$$D \simeq -2ay_r(p + f^2) \quad (46)$$

$$E \simeq -2fy_r(p - a^2) \quad (47)$$

$$F \simeq -2ay_r(p - f^2) \quad (48)$$

As the values of D, E and F are almost the same for every pair of points having $x_l \simeq -a$, we conclude that (for a single computation of α) we can allow only one pair of points with $x_l \simeq -a$ into the computation.

Case II ($x_l \simeq a$):

In this case the point in space can be virtually anywhere in space, and, therefore, x_r can almost vary from $-a$ to a. If $x_{1r} \simeq x_{2r} \simeq a$ (points 3 and 4 in Figure 2), then it is easy to check that D and F will have the same magnitude as those in (46) and (48) but with a different sign, and E will be the same as that in (47). For small values of α, the term $E \sin 2\alpha$ will be insignificant, and the set will be the opposite (not really different) of that for the previous case. Therefore, when α is small the two pairs of points in Figure 2 should not be combined.

If $x_{1r} \simeq x_{2r} \simeq -a$ (pair 5,6 in Figure 3), then

$$D \simeq 2ay_r(p - f^2) \quad (49)$$

$$E \simeq -2fy_r(p + a^2) \quad (50)$$

$$F \simeq 2ay_r(p + f^2) \quad (51)$$

This set is substantially different from the previous ones, indicating that such a pair of points goes well with either the pair (1,2) or the pair (3,4) in Figure 2. Also it is easy to verify that a pair of points with $x_{1r} \simeq -x_{2r}$ (pair 7,8 in Figure 4) appears to go well with all three previous pairs. The concern, however, is that pairs of points such as those illustrated in Figures 3 and 4 are relatively rare, and often we may not find any. Pairs of points of the types (1,2) and (3,4) are more likely to be available, but when α is small we cannot rely on them.

Generally speaking, on many occasions we may not be able to compute α with a given degree of reliability. The question is what to do when this happens. Previously it has been shown that the recovery of points in space, while quite sensitive to error in θ, is relatively insensitive to error in α. That is to say, it is the dependence of θ on α which makes the knowledge of the accurate value of α of special importance. The point, however, is that the degree of this dependence depends on the image points that we choose. An interesting question comes to mind: Is it possible to find a pair of points such that the computed value of θ becomes sufficiently insensitive to small variations in the value of α? In general, for a single pair of points, the answer is no. However, in the next section we will see that if we use the two pairs of points shown in Figure 2, then the average of the two computed values of θ will generally be insensitive to small changes in α, providing us with a reliable estimate of θ despite the relative uncertainty in the value of α.

5. COMPUTATION OF RELATIVE PAN ANGLE WITHOUT WORLD PAN ANGLE

We would like to explore the possibility of computing θ when the value of α is not known accurately. Suppose we have two points that not only pair well (for computation of $d\theta$), but also have their x_r's roughly equal; namely, two points for which we have the following relations:

$$x_{1r} \simeq x_{2r} \tag{52}$$

$$x_{1l} \simeq x_{2l} \tag{53}$$

$$y_{1l} \simeq -y_{2l} \tag{54}$$

(Examples of such pairs are points (1,2) and (3,4) in Figure 2.). Using equations (A11) and (A6) in Appendix A and approximating a_{1r} and a_{2r} by f, the above relations allow us to write

$$x_{1r}\gamma_2 - x_{2r}\gamma_1 \simeq x_r(\gamma_2 - \gamma_1) \simeq f x_r(\delta_{2y} - \delta_{1y})\cot\alpha - 2x_r y_r b_l\theta/\sin\alpha \tag{55}$$

$$f(\delta_{2y} - \delta_{1y}) \simeq -2y_r\delta_z\sin\alpha + 2y_r b_l\theta \tag{56}$$

where x_{1r} and x_{2r} are denoted by x_r, y_{1r} and y_{2r} by y_r and b_{1l} and b_{2l} by b_l.

The above equation indicates that for negligible values of θ the difference in vertical disparities vanishes as δ_z does. It must be emphasized that without the above mentioned assumptions the expression for the difference of vertical disparities involves terms that will not vanish as $\delta_1 x$ does (or $\delta_1 x$ and $\delta_2 x$ do). This can be easily verified from equation (A6).

Employing equations (55) and (56) the coefficients A through F for the case $\theta_0 = 0$ may be written as

$$A \simeq -2y_r(p\sin\alpha + f x_r\cos\alpha)\delta_z + 2y_r b_l(p - f x_r\tan(\alpha/2))\theta \tag{57}$$

$$B \simeq -2y_r(f x_r\sin\alpha - p\cos\alpha)\delta_z + 2y_r b_l(f x_r + p\tan(\alpha/2))\theta \tag{58}$$

$$C \simeq -2y_r(p\sin\alpha - f x_r\cos\alpha)\delta_z + 2y_r b_l(p + f x_r\tan(\alpha/2))\theta \tag{59}$$

$$D \simeq 2y_r[p(x_r - \delta_z) + f^2 x_r] \tag{60}$$

$$E \simeq 2y_r f[x_r(x_r - \delta_z) - p] \tag{61}$$

$$F \simeq 2y_r[p(x_r - \delta_z) - f^2 x_r] \tag{62}$$

where p denotes $p_1 \simeq p_2$.

It is worth mentioning that numerical experiments have shown, even for cases when the relations (52) through (54) are only approximately valid, that the above expressions provide fairly good approximations to the exact values of the six coefficients.

To evaluate the dependence of θ on α, we differentiate equation (28) with respect to α to obtain

$$d\theta/d\alpha = \frac{(BD - AE) + (CD - AF)\sin(2\alpha) + (BF - CE)\cos(2\alpha)}{(D\cos(2\alpha) + E\sin(2\alpha) + F)^2} \tag{63}$$

From equations (57) through (63) we can make the following observations:

a) For negligible values of θ, as A, B and C go to zero as δ_z does, so will all the terms in the numerator of $d\theta/d\alpha$. Therefore, for a pair of points that satisfy the relations (52) through (54), and are deep in space (small δ_z), the computed value for θ will be fairly insensitive to the inaccuracy in α. In practice, however, unless δ_z is extremely small and θ is no larger than a degree or so, the computed value of θ (when α is inaccurate) may not be accurate enough.

b) Let us denote the set of six coefficients that results from pairing points 1 , 2 in Figure 1 by $A_{12}, B_{12}, \ldots, F_{12}$, and the set resulting from points 3 , 4 (in the same Figure) by $A_{34}, B_{34}, \ldots, F_{34}$. In equations (57) through (62), if α is small and $|x_{1r}|$ is large, then terms involving $\sin\alpha$ and $\tan(\alpha/2)$ may be ignored and regardless of the magnitude of θ the following relations will emerge:

$$A_{12} \simeq -A_{34}, \quad B_{12} \simeq B_{34}, \quad C_{12} \simeq -C_{34} \tag{64}$$

$$D_{12} \simeq -D_{34}, \quad E_{12} \simeq E_{34}, \quad F_{12} \simeq -F_{34} \tag{65}$$

$$B_{12}D_{12} - A_{12}E_{12} \simeq -(B_{34}D_{34} - A_{34}E_{34}) \tag{66}$$

$$B_{12}F_{12} - E_{12}C_{12} \simeq -(B_{34}D_{34} - E_{34}C_{34}) \tag{67}$$

With the terms $(DC - AF)\sin 2\alpha$ and $E\sin 2\alpha$ in the equation (63) being small, we conclude that

$$d\theta_{12} \simeq -d\theta_{34} \tag{68}$$

Denoting the true value of θ by θ_0, and those computed by pairs 1,2 and 3,4 by θ_{12} and θ_{34}, respectively, we have:

$$\theta_0 \simeq \theta_{12} + d\theta_{12} \tag{69}$$

and

$$\theta_0 \simeq \theta_{34} + d\theta_{34} \tag{70}$$

Therefore

$$\theta_0 \simeq (\theta_{12} + \theta_{34})/2 \tag{71}$$

c) When α is not small equations (64) and (65) will not be satisfied. Therefore, we should not simply take the average of the two computed values of θ. What we should do instead is to take a weighted average of θ_{12} and θ_{34}. For example, if $|d\theta_{12}| < |d\theta_{34}|$, then θ_0 will be closer to θ_{12}, and therefore θ_{12} should be assigned a larger weight. It is noted that although we can usually calculate $d\theta_{12}$ and $d\theta_{34}$ rather reliably, computational experiments have shown that, in general, we should not estimate θ_0 from $\theta_{12} + d\theta_{12}$ or $\theta_{34} + d\theta_{34}$. As was suggested earlier, a much better approach is to use $d\theta_{12}$ and $d\theta_{34}$ for assigning appropriate weights to θ_{12} and θ_{34}.

Note, however, that the calculation of the (weighted) average of θ_{12} and θ_{34} will make sense only so long as $d\theta_{12}$ and $d\theta_{34}$ are of opposite signs. Obviously for small values of α, $d\theta_{12}$ and $d\theta_{34}$ do have opposite signs. We should therefore find out what happens when the magnitude of α increases. This is done in Appendix C. The conclusion is that within the range $-20 < \alpha < 20$ degrees $d\theta_{12}$ and $d\theta_{34}$ have different signs, and calculation of a weighted average is possible. Since the two pairs of points (1,2) and (3,4) in Figure 2 cannot be combined for the computation of $d\alpha$ only when $|\alpha|$ is small (say less than 7 to 8 degrees), therefore the range $-20 < \alpha < 20$ is wide enough.

d) The case when θ_0 is not small can be analyzed in the same manner as was done in the previous case.

6. COMPUTATION OF RELATIVE TILT ANGLE

For the case $\alpha = \theta_0 = 0$ equation (8) reduces to

$$\phi = \frac{f[y_{1r}x_{1l}(y_{2r} - y_{2l}) - y_{2r}x_{2l}(y_{1r} - y_{1l})]}{x_{1l}y_{1r}(f^2 + y_{2r}y_{2l}) - x_{2l}y_{2r}(f^2 + y_{1r}y_{1l})} \qquad (72)$$

It can be seen that, in general, so long as vertical disparity can be calculated reliably so can be the relative tilt angle. In particular, pairs of points recommended for the computation of θ seem to be quite appropriate for the computation of ϕ too.

Even for large values of α and θ, the computation of ϕ appears straightforward and stable. In fact we only have to pick pairs of points so that neither vertical disparities nor the denominator for ϕ (which is the same as the denominator for θ) is small. For the effect of a large world pan angle on vertical disparity, δ_y, see Section 4.

Note that to calculate the world pan angle, α, instead of using the expression for θ (see equation (28)), we could have used the expression for ϕ. The reason we did not do this is because the latter approach has been found (by means of computational experiments) to be less reliable. Apparently, the reason is the weaker coupling of ϕ and α (as compared to θ and α).

7. A SCHEME FOR A RELIABLE CALIBRATION

While computation of ϕ is rather easy, computation of θ and, in particular, α is difficult. If a sufficient number of points that are appropriate for computation of these angles is available, then all of them can be computed reliably. However, in many images we may not find enough points or pairs of points that can be sufficiently trusted. Therefore, a program which is devised for the computation of pan and tilt angles should adopt the following strategy.

The program should first consider points (and pairs of points) that appear reliable. If there exist enough such points (usually about six to eight pairs, according to simulation), then points that appear less reliable need not be considered. Otherwise, those points and combination of points that do not appear as promising should be considered too. However, the program should have special code to estimate the reliability of all sensitive mathematical operations. If an operation appears unreliable, then the point (or the combination of points) must be rejected. If the reliability of a given operation is such that neither rejection nor "full" acceptance seems appropriate, then an appropriate weight, i.e. a number less than one, should be assigned to the result of that computation. Therefore, at the end, when we average over all non-rejected results, the contribution of an individual result (to the final result) will be proportional to its degree of reliability. If the sum of the weights of the individual results for a given angle is less than a certain threshold, then the computation of that angle must be considered unreliable. Although for α this may happen rather frequently, we believe that for most images computation of the more important angles, i.e. ϕ and θ will be reliable.

8. CONCLUDING REMARKS

We have studied difficulties due to digitization error which arise in computation of the relative pan and tilt angles and the world pan angle (also referred to as gaze angle). If the gaze angle is zero and the vergence angle is small, then the best points to be used in the computation of the relative orientation angles are corner points and that the best combination of points are those that pair points from two corners that have the same latitude. (In particular, points coming from two opposite corners must not be combined.) If either the world pan angle or the vergence angle is large, then of the two pairs of corners that have the same latitude only one pair of corners should be considered. Indeed, in this case points that are horizontally near the middle of the image may give rise to more stable computations than those from the "bad" corners.

Since the computation of the world pan angle is frequently unreliable, it is necessary to bypass the computation of this angle and to compute the relative orientation angles. This can often be done; despite the fact that in the analytic formulation of the problem the relative orientation angles and the world pan angle are coupled. If the world pan angle cannot be computed reliably, one should take the average of results obtained from the two pairs of corners (unless the horizontal disparities of these points are considerably different from each other). When the world pan angle or the vergence angle are large, then to take average, points from the inappropriate corner should be replaced by points closer to the middle of the image. Also, simple averaging should be replaced by a weighted averaging. It is interesting to note that Kamgar-Parsi and Eastman [7] have shown that to compute the relative roll angle when the value of the gaze angle is not known accurately, one should take the average of results obtained by combining points from two different sets of three corners.

APPENDIX A

Numerical computation of θ, ϕ and α requires the calculation of many intermediate quantities (such as γ, δ, etc.). To discuss the computational difficulties, we have to examine the magnitudes of these quantities separately. In the main text these quantities are expressed in terms of x_r, x_l, y_r and y_l (of a given point in space), which makes it difficult to estimate their magnitudes. This is because for a point in space not all x and y coordinates of both left and right image positions are independent quantities. Therefore in this Appendix we have eliminated either y_l or y_r or x_r and have derived expressions for the intermediate quantities in

terms of θ, ϕ and the remaining projected quantities.

First, we shall derive expressions for $1/\lambda_l$ and λ_r/λ_l. Equations (2) and (4) in the main text can be written as

$$x_r(\lambda_r/\lambda_l) = d_l - c_l\theta + y_l \sin\theta_0\phi + b\cos\alpha/\lambda_l \quad \text{(A1)}$$

$$-f(\lambda_r/\lambda_l) = -c_l - d_l\theta + y_l \cos\theta_0\phi + b\sin\alpha/\lambda_l \quad \text{(A2)}$$

Elimination of λ_l except when it occurs in ratio with λ_r yields

$$\lambda_r/\lambda_l = \frac{a_l + b_l\theta - y_l\cos\alpha\phi}{a_r} \quad \text{(A3)}$$

Expression for y_l:

Equations (A3) and (3) yield

$$y_l = \frac{y_r a_l' + y_r b_l'\theta - f a_r\phi}{a_r + y_r(\cos(\theta_0+\alpha)\phi} \quad \text{(A4)}$$

Expression for y_r:

Equation (A4) can be written as

$$y_r = \frac{a_r(y_l + f\phi)}{a_l' + b_l'\theta - y_l\cos(\theta_0+\alpha)\phi} \quad \text{(A5)}$$

Expression for Vertical Disparity when $\theta_0 = 0$:

From equation (A4) we obtain

$$\delta_y = y_r - y_l = \frac{y_r\delta_z\sin\alpha + (fa_r + y_r^2\cos\alpha)\phi - y_r b_l\theta}{a_r + y_r\cos(\alpha)\phi} \quad \text{(A6)}$$

Note that the above expression does not involve y_l. Likewise we can eliminate y_r and obtain a similar expression for vertical disparity. For this, we rearrange the above equation to obtain

$$\delta_y = y_r - y_l = \frac{y_l\delta_z\sin\alpha + (fa_r + y_l^2\cos\alpha)\phi - y_l b_l\theta}{a_l + b_l\theta - y_l\cos\alpha\phi} \quad \text{(A7)}$$

where $\delta_z = x_r - x_l$ is the horizontal disparity.

Expression for x_r (when $\theta_0 = 0$ but α is not zero):

Equation (A3) can be rearranged to obtain

$$x_r = \frac{x_l y_r\sin\alpha + f\delta_y\cos\alpha + y_r b_l\theta - (f^2 + y_r y_l)\cos\alpha\phi}{(y_l + f\phi)\sin\alpha} \quad \text{(A8)}$$

Expression for $\gamma\sin\alpha - f\delta_y\cos\alpha$:

Substitution of equation (A4) into the expression of interest yields

$$\gamma\sin\alpha - f\delta_y\cos\alpha = \frac{y_r a_r b_l\theta - (fa_r^2 + y_r^2 a_l\cos\alpha)\phi}{a_r + \cos\alpha y_r\phi} \quad \text{(A9)}$$

Expression for num:

After substituting equation (A4) for y_l into the numerator of (7) and ignoring terms of the second order in θ and ϕ, we obtain

$$num = |f(y_{1r}b_{1l}a_{2r} - y_{2r}b_{2l}a_{1r}) + y_{1r}y_{2r}(y_{2r}b_{1l}a_{2l}/a_{2r} - y_{1r}b_{2l}a_{1l}/a_{1r})|\theta \quad \text{(A10)}$$

Expression for γ (in terms of x_l, y_r and y_l):

Substitution of (A8) into γ yields

$$\gamma = \frac{fy_l\delta_y\cos\alpha - (py_l\cos\alpha + fy_r x_l\sin\alpha)\phi + y_r y_l b_l\theta}{(y_l + f\phi)\sin\alpha} \quad \text{(A11)}$$

APPENDIX B

We would like to make an estimate of the error we would incur in the computation of θ for the case α=0.

The pair of points used for the computation of θ are assumed to be near two corners of the left image plane that have the same latitude. Assuming that the error in the computation of the vertical disparity is negligible, then virtually all the error will come from the computation of *num*. Utilizing the numerator in equation (21) which is essentially the same as *num*, it can be verified that for the pair of points that we have considered we can roughly write

$$num \simeq (f^2 + y_{1r}y_{1l})[(y_{2r} - y_{2l}) - (y_{1r} - y_{1l})] \quad \text{(B1)}$$

Thus, the sensitivity of *num* to error is basically the same as the sensitivity of the difference of the two vertical disparities. From equation (A7) we find

$$(y_{2r} - y_{2l}) - (y_{1r} - y_{1l}) \simeq 2x_{1l}y_{1r}\theta/f \quad \text{(B2)}$$

In units of pixels, the right hand side of the above expression is about 4θ (where θ is in degrees).

The average error for the difference of vertical disparities (in units of pixels) is equal to

$$\int_{-1/2}^{1/2}\int_{-1/2}^{1/2}\int_{-1/2}^{1/2}\int_{-1/2}^{1/2} |x_1 + x_2 - x_3 - x_4| dx_1 dx_2 dx_3 dx_4$$

Using the relation

$$\int_{-1/2}^{1/2} |x - a| dx = \begin{cases} |a| & \text{if } |a| \geq 1/2 \\ a^2 + 1/4 & \text{if } |a| \leq 1/2 \end{cases} \quad \text{(B3)}$$

after considerable algebra, the quadruple integral is found to be equal to 7/15. Therefore, on the average, the relative error in the computation of the difference of vertical disparities is $7/(60\theta)$. From this we conclude that, on the average, the absolute error in the computation of θ is $7/60 \simeq .12$ degrees.

APPENDIX C

We want to know for what values of the world pan angle, α, the derivative of the relative pan angle, θ, with respect to α, i.e. $d\theta/d\alpha$ has opposite signs for the two pairs of points (1,2) and (3,4) in Figure 2. Since we know that for small values of α, $d\theta_{12}/d\alpha < 0$ and $d\theta_{34}/d\alpha > 0$, what we want to find out is the maximum value of $|\alpha|$ for which we still have $d\theta_{12}/d\alpha < 0$ and $d\theta_{34}/d\alpha > 0$.

To do this we have to examine the magnitudes of the

terms in the numerator of $d\theta/d\alpha$ (see equation (57)), and therefore the coefficients A through F. It can be verified, comparatively speaking, that the magnitudes of D and E are "large", of B and F "medium", and of A and C (depending on the sign of x_r and the value of α) "small" to "medium". Therefore, when determining the sign of $d\theta/d\alpha$ the terms that need be considered are BD, AE (when A is not small) and $DC\sin 2\alpha$ and $EC\cos 2\alpha$ (when C is not small). Due to symmetry we only need to study the case of positive values of α. For $\alpha > 0$, depending on the sign of x_r we will distinguish two cases:

I) $\alpha > 0, \quad x_r > 0$

In this case C is small. Therefore, we only need to examine the sign of the expression $BD - AE$. Since in this case we have $A > 0, B < 0, D > 0$ and $E < 0$, both BD and AE are negative. For small values of α, $|B|$ is appreciably larger than A. Thus $BD - AE > 0$. However as α increases, B decreases and A increases. Since the magnitudes of D and E are roughly equal, $BD - AE$ changes its sign when $|A| \simeq |B|$. To find an estimate of the angle, α_c, at which this happens we set $A = -B$ to obtain

$$\tan \alpha_c = \frac{p - fx_r}{p + fx_r} \qquad \text{(C1)}$$

For a pair of points such as (3,4) in Figure 2, we have $\tan\alpha_c \simeq .5$, and therefore $\alpha_c = 27$ degrees. That is, up to about 27 degrees $BD - AE$ retains the sign it has at $\alpha = 0$. Numerical computations, however, suggest that the derivative changes its sign at about 22-24 degrees (depending on the closeness of the pair of points to the corners). Owing to the fact, unlike our assumption when we calculated α_c, that D and E are not exactly equal and also the fact that other terms can have some small effects, the discrepancy between 27 and 22-24 degrees is not unexpected.

II) $\alpha > 0, \quad x_r < 0$

In this case A is small. Therefore, we should examine the sign of the expression $BD + CD\sin 2\alpha - CE\cos 2\alpha$, or (since in this case $D < 0$ and $E < 0$) basically the expression $-B + C\cos 2\alpha - C\sin 2\alpha$. At 27 degrees the magnitudes of B and C become equal. Thus, for the term $C\sin 2\alpha$ to overcome $-B + C\cos 2\alpha$, α must be considerably larger than 27 degrees. That is, for a pair of points such as (1,2) in Figure 2, $d\theta_{12}/d\alpha$ remains positive much longer than $d\theta_{34}/d\alpha$ remains negative. Therefore, $d\theta_{12}/d\alpha$ and $d\theta_{34}/d\alpha$ have different signs up to 27 (or as was mentioned above 22-24) degrees. Since non-negligible values of θ, in some cases, may slightly decrease this range, a more dependable range will roughly be $-20 < |\alpha| < 20$ degrees.

ACKNOWLEDGEMENT

The author would like to thank Professor Larry Davis and Professor Azriel Rosenfeld for their helpful comments and discussion.

REFERENCES

[1] F. Solina, "Errors in Stereo Due to Quantization", MS-CIS-34, Department of Computer and Information Science, University of Pennsylvania (1985).

[2] R. Duda and P. Hart, *Pattern Classification and Scene Analysis*, John Wiley and Sons (1973).

[3] A. Kak, "Depth Perception for Robots", *Handbook of Industrial Robotics*, Editor: S. Nof, John Wiley and Sons (1985).

[4] R. Schalkoff, "Automatic Recalibration of Moving Cameras in Stereo Vision Systems", *Image and Vision Computing* Vol. 3, 118–121 (1985).

[5] B. Kamgar-Parsi, "Practical Computation of Pan and Tilt Angles in Stereo", University of Maryland, Center for Automation Research Technical Report 195, 1986.

[6] B. Kamgar-Parsi and B. Kamgar-Parsi, "Evaluation of Digitization Error in Computer Vision", University of Maryland, Center for Automation Research Technical Report, in preparation.

[7] B. Kamgar-Parsi and R. D. Eastman, "Calibration of a Stereo System with Small Relative Angles", University of Maryland, Center for Automation Research Technical Report 240, 1986.

Uncertainty Analysis of Image Measurements *

M. A. Snyder
Computer and Information Science
Univ. of Mass.
Amherst, Ma. 01003

Abstract

In this paper we discuss the implications for "low-level" computer vision of the uncertainty in image measurements. We illustrate these ideas with applications to 1) the recovery of depth from both stereo and motion, 2) the refinement of depth maps in multiple-frame motion algorithms, 3) the detection of independently moving objects, and 4) the relative efficacy of stereo and motion in recovering depth. We conclude with several numerical examples illustrating this analysis, and some conjectures on possible future applications of uncertainty analysis in computer vision.

1 Introduction

The calculation of environmental (3D) parameters is a major concern of many techniques in computer vision. For instance, an algorithm for stereopsis computes the 3D depth of environmental points, and a motion algorithm computes depth and/or motion parameters (rotational and translational velocities). These environmental parameters might be desired quantities in themselves, or they could be used for further processing, such as for shape recovery and the segmentation of the environment into distinct objects, or for recognition, path-planning, etc. All such techniques must take as input the position in the image or images of particular "interesting" image structures such as points, contours, and regions. In the case of correspondence-based techniques for stereo or motion, the position of corresponding points in two or more images serves as the input to an algorithm, the output of which is some environmental parameter or parameters. We will be interested here only in the case where the output is a numerical quantity, such as depth. We think that similar considerations will hold for non-numerical output quantities zero-crossings, straight lines, regions, etc., but the application of our ideas to these is not yet clear.

*This research was supported by Army ETL under grant DACA76-85-C-0008

In most correspondence-based techniques the positions of corresponding points in two views are determined either by using an interest operator (such as Moravec's [Mora1980] or Kitchen and Rosenfeld's [Kitc1982]) to select distinguished image points which are most easy to match, or by using a correlation measure. We investigate in this section how accurately the position of such image points can be determined. We confine our discussion to interest points selected by some interest operator, but identical considerations will hold for techniques which use a correlation measure.

Techniques which select interest points consist in optimizing some numerical measure. The location of the interest point is then defined as the center of the pixel in which the position of the optimum lies. It then follows that if all that is known about the interest point is that the optimum lies within a particular pixel, the actual position of the optimum is *uncertain*—it could be anywhere inside the indicated pixel. We shall express this by saying that the position of the interest point is known only to lie within an *uncertainty region* $\mathcal{R}$, which in this case is the pixel which contains the optimum. This uncertainty will be called *discretization uncertainty*, since it arises from the discretization of the image plane.

In general, image points may not be in the place predicted by the laws of projection because of sensor distortion and aliasing, in addition to the above-mentioned discretization of the image plane, or they may appear with intensity different from the laws of physics because of sensor noise and the quantization of grey-levels. These phenomena give rise to what are usually called "errors," since they lead to image structures which do not correspond to physical structures in the environment. These "errors" have no meaning in terms of the image, of course, since their existence can be ascertained only by comparison of the image with the 3D environment which gave rise to it. Since this comparison is ordinarily not available for a vision system, the concept of "error" is not generally useful.

Although "errors" cannot be experimentally defined in terms of the image alone, the *effects* of errors can be so defined. Let us represent the interest operator as a surface, with the optimum corresponding to a peak in the surface. We shall call the position in the image of this optimum the *nominal* position of the interest point in question. Er-

rors of the sort noted above will generally lead to surfaces which have a broad peak, rather than the sharp peak that would be obtained in their absence. It is intuitive that the broader the peak, the greater must have been the effect of "errors" (whatever their source) on the position of the interest point, and the less certain we should be that the position of the peak corresponds to the position that would be found in the absence of any error. That is, the *uncertainty* in the position of the peak should be related to the broadness of the peak; a broad peak corresponds to a more uncertain position of the peak than does a narrow peak. If this uncertainty is ignored, the position of the interest point will be what we have called the "nominal" position of the interest point. We will therefore take the adjective "nominal" to mean "in the absence of uncertainty." The nominal value of a quantity **Q** will be denoted by enclosing it in angle brackets: $\langle Q \rangle$.

We make these observations more quantitative by defining the ***uncertainty region*** (UR) for the interest point as the largest connected set of image points in the neighborhood of the position of the optimum of the interest measure for which the value of the interest measure exceeds a certain threshold θ. The dependence of this uncertainty region on the threshold is a reflection of the fact that there is no canonical definition of the "broadness" of the peak. A more precise definition could be given by fitting a Gaussian surface to the interest measure surface, and defining the UR in terms of the principal curvatures (or standard deviations along the principal directions) of the surface. Although this would be a more mathematically satisfying procedure, it would also clearly be more computationally expensive. We therefore view the proper definition of the uncertainty region for each interest point to be an experimental question; we will not consider it further in the present work. If a quantity **Q** is a scalar quantity (such as depth), the UR is just an interval I, which we will write as

$$I = [Q^{\text{mean}} - \delta Q, Q^{\text{mean}} + \delta Q]. \tag{1}$$

The fact that the value Q is in this interval will also be expressed by the standard notation

$$Q = Q^{\text{mean}} \pm \delta Q. \tag{2}$$

Here Q^{mean} is just the center of the uncertainty interval, and δQ is the uncertainty in **Q**.

Since errors are not defined in terms of the image alone, they are not sources of information about the image. However, the structure of the interest measure around an interest point is information about the *effect* of errors on the image. Indeed, it is *all* of the information that can be obtained about these errors from the image alone. Consequently, the calculation of uncertainties can be viewed as a conversion of "non-information" (errors) into information. This information is not ordinarily used. The most important point of the present work is that if one truly wishes to use all the information present in an image, then the uncertainty in the positions of interest points (or any other directly measured quantity) *must* be calculated.

The definition of the uncertainty region for an interest point F is an essentially mathematical one, but should have some physical interpretation. We propose that the uncertainty region should be chosen so that it is "highly likely" that the interest point in fact lies somewhere inside the uncertainty region. This means that we make the following physical interpretation of the uncertainty region: *any image point in the uncertainty region for the interest point F is a possible position of the interest point* consistent with the information in the image. This means that although in a particular experiment (i.e., a particular image) we find a definite position for the interest point (namely, its nominal position), any other point in the uncertainty region *could* have been found as the position of the interest point (because of the random nature of sensor noise, phase effects in aliasing, etc.

1.1 Combining Uncertainties

A vision algorithm will generally directly measure some quantities, such as the positions and grey-level intensities of image points, and then use those directly measured quantities to compute other (*derived*) quantities, such as depths or motion parameters. We discuss here how the uncertainty regions for directly measured quantities are used to compute the uncertainty regions for derived quantities.

Suppose that $\mathbf{P}_1$ and $\mathbf{P}_2$ are two directly measured quantities, and that **Q** is a quantity which is computed from $\mathbf{P}_1$ and $\mathbf{P}_2$. That is, there is some functional relation between the values each quantity takes:

$$f(P_1, P_2, Q) = 0. \tag{3}$$

Suppose that we have determined the uncertainty regions $\mathcal{R}_1$ and $\mathcal{R}_2$ for P_1 and P_2. Pick a point P_1 in $\mathcal{R}_1$ and a point P_2 in $\mathcal{R}_2$ and compute, from (3), the resulting value of Q. If P_1 and P_2 are then allowed to independently range over their respective uncertainty regions, Q will range over some region $\mathcal{R}$. Owing to the physical interpretation we gave to the uncertainty region, this region is just the uncertainty region for the derived quantity **Q**. The calculation of the uncertainty regions for the quantities computed by an algorithm will be called an ***uncertainty analysis*** (UA) of the algorithm.

We recall that the nominal value $\langle Q \rangle$ for a quantity **Q** is the value that **Q** would take if uncertainty were neglected. The value Q^{mean}, on the other hand, is just the center of the uncertainty interval for **Q**. For directly measured quantities, we will (for simplicity only) in general take these two values to be equal. For derived quantities, however, there is no *a priori* reason for the equality of the "mean" and nominal values. Indeed, in section 3 we will see examples where these two values differ.

1.2 Uncertainty regions and probability measures

An alternative approach to the analysis of uncertainty can be given by ascribing a probability to each value that a quantity **Q** takes. This point of view has been taken by several researchers [Faug86,Matt86]. This would be implemented by giving the probability density $\rho(Q)$, where $\rho(Q)\,dQ$ is the probability that **Q** takes a value between Q and $Q + dQ$. One could assume, as in photogrammetry, [Slam80] that (in the absence of evidence to the contrary) $\rho(Q)$ is a Gaussian normal distribution centered at Q^{mean}, of standard deviation σ. If **Q** is a scalar quantity this will take the form

$$\rho(Q) = \frac{1}{\sigma\sqrt{2\pi}}\exp\left[\frac{-(Q - Q^{\text{mean}})^2}{2\sigma^2}\right]. \qquad (4)$$

The distinction between the this "probability" approach and the "uncertainty region" approach is artificial, since the UR approach can be given a probability interpretation by letting the probability density be constant inside the uncertainty region and zero outside the region. For instance, if we write the uncertainty region as the interval (1), then the corresponding probability approach is to take

$$\rho(Q) = \frac{1}{2\delta Q} \qquad (5)$$

for $Q \in I$, and 0 otherwise. A rough comparison of the UR approach with the Gaussian normal approach can be obtained by taking $\delta Q = \sigma$.

For simplicity, we have ignored all probability considerations in defining the uncertainty regions for derived quantities. For example, although the addition of two uncertain quantities (each having the "flat" distribution (5)) will have a "triangular" probability distribution (since this is the convolution of two flat distributions), we will treat it as "flat."

Whether a "probability" approach using, for example, a Gaussian normal probability density, or an "uncertainty region" approach is most appropriate depends on the particular task at hand, the hardware on which the algorithm is implemented, etc. The probability approach has the virtue of being the most elegant and mathematically satisfactory, but it will generally involve more computation than the uncertainty region approach. We thus suspect that the probability approach will be of interest mainly in theoretical work, while the UR approach will be of more practical interest. We will use only the UR approach in what follows, but the reader should keep in mind that everything we have to say about uncertainty can be readily couched in the language of probability densities.

2 Why uncertainty analysis is important

Uncertainty analysis (UA) is important, even critical, for the performance of some common tasks of vision systems:

- **Obstacle Avoidance** A sensor must use an algorithm which computes the 3D position of objects in the environment if it is to navigate without colliding with obstacles. By performing a UA of the algorithm, one calculates the uncertainty region for each of the obstacles. This region corresponds, according to our physical interpretation, to the possible positions of the object, and is therefore the region which must be avoided by the sensor. Only a portion of these regions (that corresponding to the nominal positions of the obstacles) would be obtained if no UA of the algorithm is performed.
- **When an algorithm should not be used** If an algorithm computes the value Q of some quantity **Q**, but a UA of the algorithm shows that the relative uncertainty in Q is large (i.e., close to 100%), then the algorithm will not give accurate results, and so should not be used in cases where precision is important. For instance, highly accurate depths are required if the shape of environmental objects is to be determined from a depth map. If the algorithm in question gives highly uncertain depths, then it should not be used to compute shape. Again, without a UA of the algorithm, this could not be known.
- **Deciding between algorithms** When two (or more) vision modules (e.g., stereo and motion) compute the same environmental parameter (e.g., depth), a UA of each will give the circumstances under which each module is expected to give the most accurate results for the parameter. This can be used to decide which module should be used in order to find the most accurate values for the parameter (see section 4).
- **Calculating search regions** In the case of multiple frame motion algorithms, or any other algorithm in which the positions of image points are used to predict the search region for further instances of the point, UA can be used to constrain the search region, and hence will give a more computationally efficient algorithm (by preventing too large a search region), and will minimize the problem of false matches (by ensuring that the search region is large enough to guarantee the presence of the desired match). Absent an uncertainty analysis, these search regions can only be guessed.

- **Generality** Most vision algorithms work well only on a few images or sets of images, and poorly on anything else. One reason for this lack of generality is that the performance of the algorithm is often highly dependent on the values assigned to certain weights or thresholds. In addition, these values are often assigned by the experimenter so as to maximize this performance on a particular set of images. Since UA will give the uncertainty in the parameters of the algorithm, in certain cases the inverse of the uncertainty can be used as a weight, or to construct an appropriate threshold. UA can thus possibly be used to interactively set these weights or thresholds according to the uncertainties found by the algorithm, and hence could possibly lead to more robust general purpose vision systems. Some efforts along these lines have been made by Moravec [Mora1980] and by Canny [Cann83] (see section 3).

- **Robustness and Graceless Degradation** With few exceptions, present vision algorithms are numerically unstable and fragile. This graceless degradation is a direct consequence of the sensitivity of the algorithms to noise and other "errors." An algorithm which incorporates an uncertainty analysis, on the other hand, takes such "errors" as part of the informational input to the algorithm. It is thus possible that UA may furnish a general way of implementing the principle of graceful degradation [Marr82] directly into a computational theory.

3 Uncertainty Analysis and Motion

Uncertainty analysis has important consequences for motion algorithms. In this section we address the implications of UA for two cases of interest in motion research. We first consider a camera undergoing uniform translational motion through a rigid environment, i.e., there is only a single rigid object, namely the environment itself. In the second case, we consider the implications of UA when the environment consists of several independently moving rigid objects.

3.1 Uniform camera translation through a rigid environment

3.1.1 Basic Equations

We assume the usual geometry of perspective projection, as illustrated in Figure 1, where we have chosen the focal length of the projection to be unity. The camera coordinate system is denoted by (X, Y, Z). As mentioned in the introduction, the camera is moving with constant speed in the $+Z$ direction through a rigid environment. In this case the motion of the image point p of some 3D point P is analytically simple [Long1980]; each point p moves along a straight trajectory, and and all such straight lines intersect at a single point O. Since all image points move away from O, this point is called the *Focus of Expansion* (*FOE*).

If we let $D(t)$ be the distance of p from the FOE at time t, and let $Z(t)$ be the $Z-$coördinate of the 3D point P to which p corresponds, then it is easy to show [Long1980] that

$$\frac{D(t') - D(t)}{D(t)} = \frac{Z(t) - Z(t')}{Z(t')}, \tag{6}$$

where t and t' are any two times. We shall refer to $Z(t)$ as the *depth* of P at time t. It follows immediately from (6) that the quantity $D(t)Z(t)$ is a *constant of the motion*, i.e.,

$$D(t)Z(t) = D(t')Z(t') \tag{7}$$

for all t and t'.

We assume (without loss of generality) that the environment is sampled at regular intervals t_n, where $t_n = n\tau$ $(n = 0, 1, \ldots, N)$, and τ is the (constant) interval between successive frames. Since the motion of the camera is uniform, we let T be the distance the camera moves toward the environment in the time interval τ. We take $t' = t_{n+1}$, $t = t_n$ in (6), and define:

$$D_n = D(t_n); \quad Z_n = Z(t_n); \quad \Delta D_n = D_{n+1} - D_n$$

(see Figure 2). Then since $T = Z_n - Z_{n+1}$ for all n,

$$\frac{\Delta D_n}{D_n} = \frac{D_{n+1} - D_n}{D_n} = \frac{Z_n - Z_{n+1}}{Z_{n+1}} = \frac{T}{Z_{n+1}}. \tag{8}$$

We can then rewrite (8) as

$$D_{n+1} = K_n D_n, \tag{9}$$

where

$$K_n = 1 + \frac{T}{Z_{n+1}} = \frac{Z_n}{Z_{n+1}} = \frac{Z_n}{Z_n - T} > K_{n-1} > 1. \tag{10}$$

If we denote by p_n the position of the projected environmental point P at time t_n, then all the p_n (for $n = 0, \ldots, N$) lie along a line. Clearly, the displacements of p between frames are constrained to lie along this line, so we will call this line the *displacement path* (see Figure 2).

If we denote by $\vec{r}_j$ the vector from the FOE to p_j, $j = 0, \ldots, N$, then since all the p_j lie along the displacement path equation (9) implies that

$$\vec{r}_{n+1} = K_n \vec{r}_n. \tag{11}$$

Note that $|\vec{r}_j| = D_j$. Equation (11) will prove to be very useful in the following.

3.1.2 Definitions

Let F be an interest point in the image. We denote by F_n the instance of F in frame n at time t_n. The position of F_n will be the point $p_n = (x_n, y_n)$. The nominal value that any quantity Q takes will be denoted by enclosing Q in angle brackets: $\langle Q \rangle$. Thus, the nominal position of F_n will be denoted by $\langle p_n \rangle = (\langle x_n \rangle, \langle y_n \rangle)$. Recall that the nominal value $\langle Q \rangle$ is the value that Q would have in the absence of

uncertainty. We will refer to the line that passes through the nominal positions of the FOE, F_0, F_1, ..., F_n as the *nominal displacement path (NDP)*.

As noted in the introduction, the position of F_n is uncertain but can be assumed to be in an uncertainty region $\mathcal{R}_n$ which contains the nominal point $\langle p_n \rangle$. In order to investigate the situation analytically, we must make some assumption about $\mathcal{R}_n$. We will choose $\mathcal{R}_n$ to be a rectangle R_n centered at $\langle p_n \rangle$, with sides parallel to the $x-$ and $y-$axes of length $2\Delta x_n$ and $2\Delta y_n$, respectively. We will denote these sides as $\{2\Delta x_n, 2\Delta y_n\}$. Thus,

$$(x_n, y_n) = p_n \in R_n \iff \begin{cases} \langle x_n \rangle - \Delta x_n \le x_n \le \langle x_n \rangle + \Delta x_n, \\ \langle y_n \rangle - \Delta y_n \le y_n \le \langle y_n \rangle + \Delta y_n \end{cases} \tag{12}$$

We assume, similarly, that the location of the FOE is uncertain, but lies within a rectangle R of sides $\{2\Delta x, 2\Delta y\}$ centered at the nominal position $\langle$FOE$\rangle$ of the FOE. We choose the coördinate system such that $\langle$FOE$\rangle$ is at the origin $(0,0)$ of the coördinate system.

There are essentially two cases of interest in this analysis. In the first case we assume that the depth of the environmental point is unknown, and calculate the uncertainty in this depth which follows from the uncertainty in the positions of the interest point in the two frames (the "startup" process). In the second case we assume that the depth (and its attendant uncertainty) have been found in one of the two frames, and use this and the uncertainties in p_0 and p_1 to find the search region for the interest point in subsequent frames (the search portion of the "updating" process). We will only summarize the results; the details of all calculations in the remainder of this section can be found in [Snyd86a].

3.1.3 Calculating the uncertainty in depth which follows from uncertainly known FOE, F_0, and F_1

We assume that the FOE and the positions p_0 and p_1 of the interest point F are known only to lie inside their respective uncertainty rectangles (see Figure 3). We will use the positions and sizes of these rectangles to calculate both the "best estimate" Z_1^{mean} for the environmental depth Z_1, and the uncertainty δZ_1 in this value. We have from (8) that

$$Z_1 = \frac{D_0}{\Delta D_0} T, \tag{13}$$

where D_0 and ΔD_0 are assumed to lie in the intervals

$$\begin{aligned} \langle D_0 \rangle - \delta D_0 \le\ & D_0 \le \langle D_0 \rangle + \delta D_0, \\ \langle \Delta D_0 \rangle - \delta(\Delta D_0) \le\ & \Delta D_0 \le \langle \Delta D_0 \rangle + \delta(\Delta D_0), \end{aligned} \tag{14}$$

and T is known accurately ($\delta T = 0$, $T = \langle \mathrm{T} \rangle$). We have equated the mean and nominal values because the quantities involved are directly measured quantities. The latter assumption is made only for simplicity of exposition; the effect of uncertain T is easily included, but complicates the appearance of the formulas. We then use (13) to find the allowed range of values for Z_1:

$$Z_1^{\text{mean}} - \delta Z_1 \le Z_1 \le Z_1^{\text{mean}} + \delta Z_1.$$

To simplify the analysis, we first define the *relative uncertainty* α in D_0, and the *relative uncertainty* β in ΔD_0:

$$\alpha = \frac{\delta D_0}{\langle D_0 \rangle} \ ; \ \beta = \frac{\delta(\Delta D_0)}{\langle \Delta D_0 \rangle}. \tag{15}$$

We also define the nominal value $\langle Z_1 \rangle$ of Z_1 to be

$$\langle Z_1 \rangle = \frac{\langle D_0 \rangle}{\langle \Delta D \rangle_0} \langle T \rangle. \tag{16}$$

That is, $\langle Z_1 \rangle$ is just the value one would expect from (13) in the absence of uncertainty.

We then have that

$$\begin{aligned} Z_1^{\text{mean}} \pm \delta Z_1 &= \langle \mathrm{T} \rangle \cdot \frac{\langle \mathrm{D_0} \rangle \pm \delta \mathrm{D_0}}{\langle \Delta \mathrm{D_0} \rangle \mp \delta(\Delta \mathrm{D_0})} \\ &= \langle Z_1 \rangle \cdot \frac{1 \pm \alpha}{1 \mp \beta}, \end{aligned} \tag{17}$$

from which it follows that

$$Z_1^{\text{mean}} = \langle Z_1 \rangle \cdot \frac{1 + \alpha\beta}{1 - \beta^2}, \tag{18}$$

$$\delta Z_1 = \langle Z_1 \rangle \cdot \frac{\alpha + \beta}{1 - \beta^2}. \tag{19}$$

Note that the "best" estimate Z_1^{mean} for Z_1 is not equal to the value $\langle Z_1 \rangle$ which would be obtained by substituting the best values for the other quantities into (13). Indeed, $Z_1^{\text{mean}} \ge \langle Z_1 \rangle$, equality holding only when $\beta = 0$, i.e., there is no uncertainty in ΔD_0. The relative uncertainty in Z_1 is, using (18) and (19), given by

$$\frac{\delta Z_1}{Z_1^{\text{mean}}} = \frac{\alpha + \beta}{1 + \alpha\beta}. \tag{20}$$

At the end of this section we use the result (20) to give quantitative meaning to two well known results in motion analysis.

In practice, the relative uncertainty in inter-frame interest point displacements is often much larger than that in the distance of interest points from the FOE. If we therefore neglect α in (18) and (19),

$$Z_1^{\text{mean}} \cong \frac{\langle Z_1 \rangle}{1 - \beta^2}, \tag{21}$$

$$\delta Z_1 \cong \beta Z_1^{\text{mean}}, \tag{22}$$

and hence the relative uncertainty in Z_1 is just equal to the relative uncertainty β in ΔD_0.

The expressions for α and β in terms of image quantities depends on how the NDP passes through the uncertainty rectangles. In the case (called VVV in [Snyd86a]) that it

passes through the vertical sides of each uncertainty rectangle (see Figure 3), we find:

$$\begin{aligned}\alpha &= \frac{\Delta x + \Delta x_0}{\langle x_0 \rangle}, \\ \beta &= \frac{\Delta x_0 + \Delta x_1}{\langle x_1 \rangle - \langle x_0 \rangle} \\ &= \left[\frac{1}{\langle K_0 \rangle - 1}\right] \frac{\Delta x_0 + \Delta x_1}{\langle x_0 \rangle} \\ &= \left[\frac{1}{\langle K_0 \rangle - 1}\right] \left[\frac{\Delta x_0 + \Delta x_1}{\Delta x + \Delta x_0}\right] \alpha. \end{aligned} \tag{23}$$

Algebraically, the case VVV corresponds to the condition

$$\frac{\Delta y}{\Delta x}, \quad \frac{\Delta y_0}{\Delta x_0}, \quad \frac{\Delta y_1}{\Delta x_1} \quad > \quad \frac{\langle y_0 \rangle}{\langle x_0 \rangle}. \tag{24}$$

There are two intuitive and commonly accepted situations in motion research where depth estimates are unreliable:

- When the interest point is "close" to the FOE
- When the environmental correlate to the interest point is "far away" from the camera.

To our knowledge a serious quantitative analysis of these statements has never been provided. We can use the results of this section to do just that. In terms of the case VVV given here, and using the equations (20) and (23) we find

- The relative uncertainty in the depth is large when the interest point is "close" to the FOE, where "close" means that either $\Delta x + \Delta x_0$ or $\Delta x_0 + \Delta x_1$ is comparable to $\langle x_0 \rangle$. A sufficient, but not necessary, condition for this is that Δx_0 is comparable to $\langle x_0 \rangle$, i.e., the uncertainty in the position of the interest point is comparable to the distance of the interest point from the FOE.
- The relative uncertainty in the depth is large when the environmental correlate to the interest point is "far away" from the camera, where "far away" means that $\langle Z_0 \rangle \gg \langle T \rangle$, i.e., the camera has moved between frames only a small fraction of the distance to the point.

The reader should keep in mind, however, that equation (20) is the mathematical expression of the results above.

3.1.4 Calculating the search region for an interest point in subsequent frames

One of the most important uses of uncertainty analysis is to use the uncertainty in the position of interest points in two frames of a dynamic image sequence and in the position of the FOE to constrain the search region for the instant of the interest point in subsequent frames. The first attempt at this was made by Bharwani, Riseman, and Hanson [Bhar85]. However, their analysis was essentially one-dimensional, since they confined their search to the nominal displacement path. Although in some cases this may be sufficient, the uncertainty region for interest points is in general two-dimensional, and so a two-dimensional analysis is of interest. This analysis was made in [Snyd86a], to which the interested reader is referred for details. The general idea is to use the uncertainty regions for the *FOE*, F_0, and F_1 to compute the uncertainty interval for the depth Z_1. One then neglects the uncertainty region for F_0 (its presence will be reflected in the uncertainty interval for Z_1) and uses the uncertainty interval for Z_1 and the uncertainty regions for the FOE and F_1 to compute the search region for F_2. One finds that this search region is the intersection of the wedge lying below a line $L_{\max}$ and above a line $L_{\min}$ with a rectangle R_2^* (see Figure 4).

3.1.5 Uncertainty Analysis and multiple-frame motion algorithms

There are several implications of UA for multiple-frame motion algorithms.

- UA can be used to find, as in the previous section, the search region for the instance of an interest point in the second frame of a sequence, given the UR for the FOE and the interest point in the first frame, and the uncertainty in the depth estimate for the environmental point (obtained, for example, from laser ranging with an ERIM). Without such an uncertainty analysis, the search region can only be guessed—an unsatisfactory situation for any motion algorithm.
- If the UR for the FOE and the interest point in any two frames of a dynamic image sequence are known, then UA can be used to find both the UR for the depth estimate and the search region for the interest point in any subsequent frame. A simple consequence of this is that if an interest point has a large UR, the estimate for its depth will be highly uncertain, and the search region for further instances of the point will be large. If one is not interested in points with large uncertainty (for instance, if one is computing shape from the depth map), then it makes little sense to follow this interest point through the sequence, since it will give highly uncertain depths (and hence be of little utility), and searching for it will be very computationally expensive. By pruning such points from the set of tracked points, the computational efficiency of an algorithm will be increased. Without an uncertainty analysis, this pruning cannot be done in any sensible way.
- UA furnishes a method of refining depth maps, in the following sense. The UR for F_i and F_j will predict a search region R_k for F_k $(i < j < k)$. This search

region represents the possible positions of F_k, consistent with the properties of the previous images. Upon performing this search and finding F_k, one can then calculate the *actual* uncertainty region for F_k (recall that this is defined in terms of the interest operator). Suppose that this *observed* uncertainty region is significantly smaller than the calculated search region. We interpret this as evidence that the uncertainty regions for F_i and F_j were too large. These can be made smaller by increasing the threshold which defined the uncertainty region until the search region for F_k and the *observed* UR for F_k are more nearly equal. This shrinkage of the UR for the previous frames thus yields a less uncertain value for the depth of the point, and hence will give a smaller search region for the interest point in subsequent frames. The important thing to note about this is that the setting of the threshold by feedback from subsequent frames has the possibility of being made automatic, in the sense that it will be done based on properties of the images alone—no human interaction would be required. This may possibly lead to more general-purpose and robust algorithms.

- A measurement of the depth in any particular frame of the sequence can be considered as a measurement of the depth in any other (previous *or* subsequent) frame, since the motion of the camera is known. Consequently, finding the uncertainty in the depth in one frame will in addition give a value for the uncertainty in the depth in any other frame. Additionally, one can consider *any* pair of frames as giving a measurement of the depth in one of the pair. By taking this pair very far apart (temporally), the relative uncertainty in the depth can be decreased (cf. equation (20)). The overall effect is that in a sequence of N frames, there will be $O(N^2)$ measurements of the uncertainty in the depth in any particular frame. These uncertainties can then be statistically combined to yield an uncertainty in that depth smaller (by a factor of order N) than any of the particular measurements alone. By thus incorporating UA into a multiple-frame motion algorithm, one will necessarily acquire less uncertain depth measurements (and smaller search regions), and hopefully more accurate ones.

Some of the above suggestions have been incorporated in a refinement of the algorithm of Bharwani, Riseman, and Hanson [Bhar86]. We are presently investigating further improvements in this algorithm.

3.1.6 Minimum time intervals for the detection of motion

If the camera moves through a rigid environment, then in order to detect the relative motion between the camera and an environmental point located at $(X, Y, Z) = (d, 0, Z_0)$ at time t_0, then one must clearly wait until the image of the point has moved at least one-half pixel in the image before any motion algorithm will be able to detect the relative motion between this point and the camera. If the camera speed is v, then a simple calculation [Snyd86b] shows that one must wait until $t = t_0 + \Delta t_{\min}$, where

$$\Delta t_{\min} = \frac{Z_0}{v} \frac{1}{1 + 2S\frac{d}{Z}} \tag{25}$$

before one will see motion of the point. This therefore corresponds to the smallest time interval one can take between the frames of a dynamic image sequence if one wishes to detect the motion of this point. The quantity S is defined as

$$S = \frac{N}{2\tan[\gamma/2]}, \tag{26}$$

where N is the resolution of the image and γ is the field of view of the camera. Note that S is just the focal length of the camera, measured in pixels. It appears in (25) because the units of focal length must be converted to pixels when measuring image quantities.

As a numerical example, suppose that the camera is attached to a car moving along a straight two-lane road at $v = 10\,\text{km/hr}$, and that a stationary obstacle (such as another car) is located in the center of the opposing lane a distance $d = 3\,\text{m}$ from the translational axis of the camera, and a distance $Z = 100\,\text{m}$ away from the camera at $t = 0$. Let $N = 256$ and $\gamma = 45°$. Then we find that at time t the obstacle is a distance Z away from the camera, and that we must wait at least until time $t + \Delta t_{\min}$ in order to see the image of the obstacle move one-half pixel from where it was at time t. This is given in Table 1 for selected values of t.

The interpretation of this table is that there is no point in taking images of the environment more closely than $\Delta t_{\min}$ apart if one is interested in detecting the relative motion between the camera and this stationary object.

3.2 The detection of independently moving objects

When the assumption of a rigid environment is relaxed, and independently moving objects in the environment are allowed, the analysis of the previous sections does not strictly hold. Nevertheless, there are some general comments that one can make about the implications of uncertainty analysis for the detection of independently moving objects.

The segmentation of the environment into independently moving rigid objects can be done in either of two ways. Either points in the image are aggregated into regions of some sort based on the similarity of their 3D motion parameters [Ullm79,Roac80], or on the basis of similarity in image parameters [Adiv85]. At any rate, though, one does not expect that different points (2D or 3D) that in fact should belong to the same segment will indeed have precisely the

same values for the appropriate parameters. Consequently, in order to determine whether points or regions with "similar" parameters should be merged into a single region must

t (sec)	Z (meters)	$\Delta t_{\min}$ (sec)
0	100.0	1.23
5	79.2	0.78
10	58.3	0.43
15	37.5	0.18
20	16.6	0.04

Table 1: Minimum time intervals for detection of motion.

involve setting a threshold which expresses how close the similarity must be in order for the merging to take place. Generally, this threshold is set "by hand." But this is exactly the kind of thing that UA is supposed to deal with. We outline here a possible way of using UA in this regard. We believe that what we have to say here has implications not only for the case of motion or stereo, but in fact for any scheme in which segmentation is performed on the basis of similiarity in numerical parameters.

Suppose that the decision whether two points are part of the same "segment" is answered by performing a Hough transform using a particular relation between image points and some parametrization. If two image points are then found in the same "bucket" in the transform space, then (modulo the uncertainty due to the quantization of the transform space) they can consistently be lumped together as arising from an image structure having that particular value for the parameters. The existence of a bucket with many such points in it is then evidence that there is a significant image structure possessing parameters having values corresponding to that bucket.

In practice, the peaks in the transform space have some width (at the very least, the size of the tessellation), and so the values of the parameters corresponding to the peak must have some uncertainty as well, which will be related to the broadness of the peak. The net effect of this is that significant image structures will have parameters which are not precise, but rather must lie in some region in parameter space. This region will be the uncertainty region (in parameter space) for this particular image structure.

The decision whether two image structures having uncertainty regions (in parameter space) $\mathcal{R}_1$ and $\mathcal{R}_2$, respectively, should be merged into a single image structure can then be easily made. The two structures should be merged if their respective uncertainty regions in the parameter space overlap "significantly," and should not be merged if there is no such overlap. This is because of the physical interpretation we gave to the uncertainty region. "Significant" can be given a quantitative meaning by defining a confidence measure which expresses how much the two URs overlap. For instance, we could define

$$\text{conf}(1,2) = \frac{\text{vol}[\mathcal{R}_1 \cap \mathcal{R}_2]}{\min\{\text{vol}[\mathcal{R}_1], \text{vol}[\mathcal{R}_2]\}}, \qquad (27)$$

where $\text{vol}[\mathcal{R}]$ is the volume of $\mathcal{R}$ in parameter space. The confidence level would then be *one* if one region contains the other, and *zero* if the two regions did not intersect. This confidence level could then be thresholded to determine when the two image structures should be merged. This could perhaps aid in the efficient segmentation of images, much as the confidence measures of Ananadan [Anan86] are used to find optical flow. These ideas are currently under investigation.

We also conjecture that there is a significant connection between the uncertainty analysis suggested here and the work of Adiv [Adiv85] on inherent ambiguities in the detection of independently moving objects. This is also currently under investigation.

4 Uncertainty analysis and stereo vs. motion for depth recovery

4.1 Uncertainty analysis for stereo

Uncertainty analysis can be applied to correspondence–based stereopsis algorithms in a manner similar to that for motion algorithms. We align the stereo baseline along the x–axis of the image coördinate system, and assume, for simplicity, that the coördinate axes of the two sensors are perfectly aligned (that is, the uncertainty in these quantities is zero). Since the images of an environmental point in the two views must lie along the epipolar line, which is parallel to the x–axis, only the uncertainty in the x–coördinate of these image points will be relevant to the computation of depth. We will let the uncertainty in these positions be Δx_L and Δx_R in the left and right views, respectively, with similar notation for the nominal positions $\langle x_L \rangle$ and $\langle x_R \rangle$ of the interest point. A trivial analysis then shows that the relative uncertainty ϵ_{rt} in the depth Z of the corresponding environmental point is given by

$$\epsilon_{\text{rt}} \equiv \left[\frac{\delta Z}{Z^{\text{mean}}}\right]_{\text{rt}} = \frac{\Delta x_L + \Delta x_R}{\langle x_L \rangle - \langle x_R \rangle}. \qquad (28)$$

We see from this that for fixed positional uncertainties in the interest point, the relative uncertainty in depth is smaller, the larger is the nominal disparity $\langle \Delta \rangle = \langle x_L \rangle - \langle x_R \rangle$.

More sophisticated treatments of uncertainties in stereo have been made by Gennery [Genn80], Kamgar–Parsi [Kamg86], and Matthies and Shafer [Matt86], to which the reader is referred. I will use only the simplified analysis above in what follows.

4.2 Stereo vs. Motion for depth recovery

In section 3, I used uncertainty analysis to find the relative uncertainty ϵ_{mot} (equation (20)) in the depth of an environmental point for the case of a camera undergoing uniform translation toward a rigid environment. Using this and the result above, we can compare the efficacy of stereo and of

motion for depth recovery in this context. We only outline the results—details can be found in [Snyd86c].

We will consider only the situation where the relative uncertainty in the distance of the interest point from the FOE is negligible compared to that in the inter-frame displacement of the point. This means that we neglect α in equation (20), and so, in the case VVV considered in section 3, we have

$$\epsilon_{\text{mot}} \equiv \left[\frac{\delta Z}{Z^{\text{mean}}}\right]_{\text{mot}} = \beta = \frac{\Delta x_0 + \Delta x_1}{\langle x_1\rangle - \langle x_0\rangle}. \tag{29}$$

We define the *motion efficacy parameter* σ to be the ratio of the relative uncertainty in depth for stereo to that for motion:σ

$$\equiv \frac{\epsilon_{\text{st}}}{\epsilon_{\text{mot}}}. \tag{30}$$

Thus, for $\sigma > 1$, motion will give less uncertain depth values than stereo, while for $\sigma < 1$ exactly the reverse obtains. We obtain from equations (28,29) that

$$\sigma \cong \frac{\Delta x_L + \Delta x_R}{\Delta x_0 + \Delta x_1} \cdot \frac{\langle x_1\rangle - \langle x_0\rangle}{\langle x_L\rangle - \langle x_R\rangle}. \tag{31}$$

In many cases, the first factor will be of order unity, since the same kind of quantity (an interest operator, or a correlation measure) is being used to find the uncertaintites, so that approximately

$$\sigma \cong \frac{\langle x_1\rangle - \langle x_0\rangle}{\langle x_L\rangle - \langle x_R\rangle} \tag{32}$$

$$\cong \frac{\text{displacement of image point between frames}}{\text{stereo disparity of image point between views}} \tag{33}$$

Thus, motion gives less (more) uncertain depth measurements than stereo when the displacement of the image point between frames of the motion sequence is larger (smaller) than the disparity between the image point in the two stereo views.

We now investigate the physical circumstances under which these situations occur. We consider a physical situation in which the environmental correlate P of some interesting point p in the image is located at

$$\begin{aligned} X &= D \\ Y &= 0 \\ Z &= Z_0. \end{aligned}$$

In the case of stereo, the nominal positions of p in the two stereo views are given by

$$\langle x_R^L\rangle = S\left(\frac{D}{Z_0} \pm \frac{d}{2Z_0}\right), \tag{34}$$

where d is the stereo baseline. Consequently,

$$\langle x_L\rangle - \langle x_R\rangle = S\,\frac{d}{Z_0}, \tag{35}$$

where S is given by (26).

We have for motion that

$$\langle x_1\rangle = \langle K_0\rangle\langle x_0\rangle, \tag{36}$$

where

$$\langle K_0\rangle = \frac{\langle Z_0\rangle}{\langle Z_0\rangle - T}. \tag{37}$$

It then follows that

$$\langle x_1\rangle - \langle x_0\rangle = \frac{T}{\langle Z_0\rangle - T}\langle x_0\rangle. \tag{38}$$

Hence, using equations (35) and (37), we find that

$$\begin{aligned} \frac{\langle x_1\rangle - \langle x_0\rangle}{\langle x_L\rangle - \langle x_R\rangle} &= \frac{T}{\langle Z_0\rangle - T}\langle x_0\rangle \frac{1}{S}\frac{\langle Z_0\rangle}{d} \\ &= \frac{T}{d}\frac{\langle Z_0\rangle}{\langle Z_0\rangle - T}\frac{\langle x_0\rangle}{S} \\ &= \frac{T}{d}\langle K_0\rangle\frac{\langle x_0\rangle}{S}. \end{aligned}$$

Equation (36) can then be used to write this in the form

$$\frac{\langle x_1\rangle - \langle x_0\rangle}{\langle x_L\rangle - \langle x_R\rangle} = \frac{T}{d}\frac{\langle x_1\rangle}{S}. \tag{39}$$

We conclude from (31) that

$$\sigma = \sigma_{\text{pos}}\,\sigma_{\text{cam}}\,\sigma_{\text{peri}}, \tag{40}$$

where

$$\sigma_{\text{pos}} = \frac{\Delta x_L + \Delta x_R}{\Delta x_0 + \Delta x_1} \tag{41}$$

is the contribution to σ from the uncertainty in the image points,

$$\sigma_{\text{cam}} = \frac{T}{d} \tag{42}$$

is the contribution to σ from the camera parameters, and

$$\sigma_{\text{peri}} = \frac{\langle x_1\rangle}{S}. \tag{43}$$

We use the subscript "peri" to refer to the latter contribution because σ_{peri} depends on $\langle x_1\rangle$; the latter is a measure of how far away the image point is from the FOE, i.e., how "peripheral" the motion of the object is.

The quantity σ_{peri} is bounded from above. This follows from the obvious fact that if the image point p is to be seen in the second frame of the motion sequence, it must be within the boundary of the image. That is (recalling that we have chosen the direction of motion of the camera to be along the Z-axis, i.e., the FOE is at the origin of the image coördinate system),

$$\langle x_1\rangle \le \frac{N}{2}.$$

Recalling the definition (26) of the quantity S, we find that

$$\sigma_{\text{peri}} \le \frac{N/2}{S} = \frac{N/2}{N/(2\tan\gamma/2)}.$$

Hence,

$$\sigma_{\text{peri}} \leq \tan \gamma/2. \tag{44}$$

We therefore find that

$$\frac{\epsilon_{\text{st}}}{\epsilon_{\text{mot}}} = \sigma \leq \frac{T}{d} \left\{ \frac{\Delta x_L + \Delta x_R}{\Delta x_0 + \Delta x_1} \right\} \tan \frac{\gamma}{2}. \tag{45}$$

If the quantity on the right-hand side of this inequality is less than unity, then stereo will *always* give less uncertain depth measurements than motion. In cases where the uncertainty in the position of image points is comparable in stereo and motion, then the factor in curly brackets is approximately unity. In this case, then, stereo will always give less uncertain depth measurements than motion whenever

$$\frac{T}{d} \tan \frac{\gamma}{2} \lesssim 1. \tag{46}$$

If we define the angle θ to be the angle which the stereo baseline d subtends at a distance of one inter-frame camera translation T, i.e.,

$$\tan \theta = \frac{d}{T}, \tag{47}$$

then the condition (46) is that the angle θ be greater than or approximately equal to one-half the field of view:

$$\theta \gtrsim \frac{\gamma}{2}. \tag{48}$$

If this condition is satisfied then stereo will give less uncertain depth measurements than motion. If this condition is not satisfied, then the question of which will give less uncertain depth measurements is open, and must be addressed by consulting the expression for σ given in (40).

We summarize the results of this section:

- In general, motion will give less uncertain depth measurements than stereo when $\sigma > 1$, and stereo will give less uncertain depth measurements than motion when $\sigma < 1$, where σ is given by (40):

$$\sigma = \sigma_{\text{pos}} \frac{T}{d} \frac{\langle x_1 \rangle}{N/2} \tan \frac{\gamma}{2}.$$

- When $\sigma_{\text{pos}} \cong 1$, motion will be less (more) uncertain than stereo when the displacement of a point between two frames is larger (smaller) than the disparity of the two points between the two views.

- When $\sigma_{\text{pos}} \cong 1$, stereo will be less uncertain than motion whenever

$$\frac{d}{T} \gtrsim \tan \frac{\gamma}{2}.$$

- If motion is less uncertain than stereo for any point in the image, it is even less uncertain for more peripheral points (assuming comparable uncertainties in the positions of the corresponding image points).

4.2.1 An example

We illustrate these results with a numerical example. Suppose that we have a stationary object located at $(\delta, 0, Z)$ at time $t = 0$. We will assume that the uncertainties in the positions of interest points are comparable in stereo and motion, so that $\sigma_{\text{pos}} \cong 1$. The motion efficacy parameter is then given by

$$\sigma = \frac{T}{d} \frac{\langle x_1 \rangle}{N/2} \tan \frac{\gamma}{2}. \tag{49}$$

But

$$\langle x_1 \rangle = S \frac{\delta}{Z - T}, \tag{50}$$

and hence, using the definition of S,

$$\sigma = \frac{T}{d} \frac{\delta}{Z - T}. \tag{51}$$

We then have that $\sigma = 1$ when $Z = Z^{\max}$, where

$$Z^{\max} = T \left\{ 1 + \frac{\delta}{d} \right\}. \tag{52}$$

Consequently, $\sigma > (<) 1$ when $Z < (>) Z^{\max}$. Therefore, stereo will give less uncertain results than motion when $Z > Z^{\max}$, and motion will give less uncertain results than stereo when $Z < Z^{\max}$.

We must, of course, require that, in the case of motion, the object be visible in the second frame. This means that $Z > Z^{\min}$, where

$$Z^{\min} = T + \frac{\delta}{\tan(\gamma/2)}. \tag{53}$$

In summary, then, stereo is more accurate than motion when

$$Z > Z^{\max}, \tag{54}$$

and motion is more accurate than stereo when

$$Z^{\min} < Z < Z^{\max}. \tag{55}$$

In Tables 2 and 3, we give the values for $Z^{\min}$ and $Z^{\max}$ (in meters) for stationary objects $\delta = 2, 5$ and 10 (meters) off the camera's translational axis, and for stereo baselines of $d = 0.5$ meters (Table 2) and $d = 1.0$ meters (Table 3). We assume that the camera is moving at 10 km/hr (= 2.78 m/sec), taking one image per second, so that $T = 2.78$ m. We take the field of view of the camera to be $\gamma = 45°$.

It is clear from this example that an uncertainty analysis such as the one performed in this section can be used to decide (depending on the physical circumstances) which one of two (or more) modules should be used to most accurately recover a particular environmental parameter.

δ (meters)	Z^{min} (meters)	Z^{max} (meters)
2	7.6	13.9
5	14.9	30.6
10	26.9	58.4

Table 2: Z^{min} and Z^{max} for stereo baseline $d = 0.5$ meters.

δ (meters)	Z^{min} (meters)	Z^{max} (meters)
2	7.6	8.3
5	14.9	16.7
10	26.9	30.6

Table 3: Z^{min} and Z^{max} for stereo baseline $d = 1.0$ meters.

5 Further applications of uncertainty analysis

We have indicated in this work many possible applications of uncertainty analysis to computer vision. We would like to mention only one other possibility. In many approaches to computing the shape of environmental objects, the relative position of at least two (and usually more) environmental points is used to compute the shape of environmental structures. If, however, the uncertainty regions of these two points overlap "significantly" (a quantitative definition of this could be made along the same lines as that of the confidence level (27) in section 3), then one can argue that the two points are not, in fact, distinguishable, and hence the shape of the environmental structure *cannot be found*. We believe this observation has important implications for many "Shape from X" algorithms. This idea is currently under study.

6 Acknowledgements

I would like to thank Prof. E. Riseman for his encouragement and advice at numerous stages in this work, and Dr. P. Anandan for many helpful discussions. I also thank Dr. I. Pavlin for asking an important question.

References

[Anan86] P. Anandan, "Measuring visual motion from image sequences," Ph. D. Thesis, UMass COINS, Amherst, Ma. (Dec. 1986).

[Adiv85] G. Adiv, "Interpreting Optical Flow," Ph. D. Thesis, UMass COINS, Amherst, Ma. (Sept. 1985).

[Bhar85] S. Bharwani, E. Riseman, and A. Hanson, "Refinement of environmental depth maps over multiple frames," DARPA I. U. Workshop Proc., Dec. 1985.

[Bhar86] S. Bharwani, E. Riseman, and A. Hanson, "On using Uncertainty Analysis to compute accurate depth maps for a mobile robot," COINS Tech. Rep. (in preparation).

[Cann83] J. Canny, "Finding edges and lines in images," MIT Tech. Rep. No. 720 (1983).

[Faug86] O. Faugeras, N. Ayache, B. Faverjon, and F. Lustman, "Building visual maps by combining noisy stereo measurements," IEEE Int. Conf. Rob. and Aut. (Apr. 1986).

[Genn80] D. Gennery, "Modelling the environment of an exploring vehicle by means of stereo vision," Ph. D. Thesis, SAIL, Stanford, Cal. (June 1980).

[Kamg86] B. Kamgar-Parsi, "Practical computation of pan and tilt angles in stereo," U. Mary. CAR Tech. Rep. CAR-TR-195 (Mar. 1986).

[Kitc1982] L. Kitchen and A. Rosenfeld, "Grey-level corner detection," Patt. Rec. Lett. **1** (1982), 95-102.

[Long1980] H. C. Longuet-Higgins and K. Prazdny, "The interpretation of a moving retinal image," *Proc. Roy. Soc. Lon.* **B208** (1980), 385-397.

[Marr82] D. Marr, *Vision*, W.H.Freeman, San. Fran., Ca. (1982).

[Matt86] L. Matthies and S. Shafer, "Error modelling in stereo navigation," CMU Tech. Rep. CMU-CS-86-140 (1986).

[Mora1980] H. P. Moravec, "Obstacle avoidance and navigation in the real world by a seeing robot," Ph.D. Thesis, SAIL, Stanford, Cal. (Sept. 1980).

[Roac80] J. Roach and J. Aggarwal, "Determining the movement of objects from a sequence of images," IEEE Trans. PAMI, vol. PAMI-2, 554-562, (Nov. 1980).

[Slam80] C. Slama, ed. *Manual of Photogrammetry*, Am. Soc. of Photog., Falls Church, Va. (1980).

[Snyd86a] M. A. Snyder, "The Accuracy of 3D parameters in Correspondence-based Techniques," UMass COINS Tech. Rep. 86-28, June 1986.

[Snyd86b] M. A. Snyder, "The detection of moving objects," UMass COINS Tech. Rep. in preparation.

[Snyd86c] M. A. Snyder, "The relative accuracy of stereo and motion for depth recovery," UMass COINS Tech. Rep. in preparation.

[Ullm79] S. Ullmann, *The Interpretation of Visual Motion*, MIT Press, Camb., Ma. (1979).

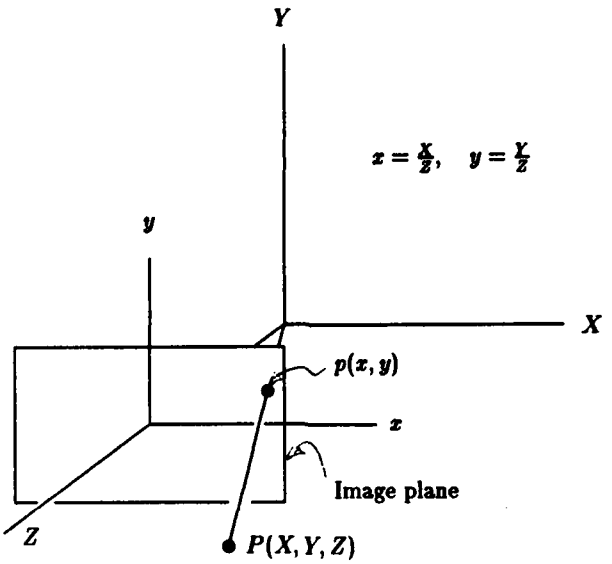


Figure 1: Perspective projection

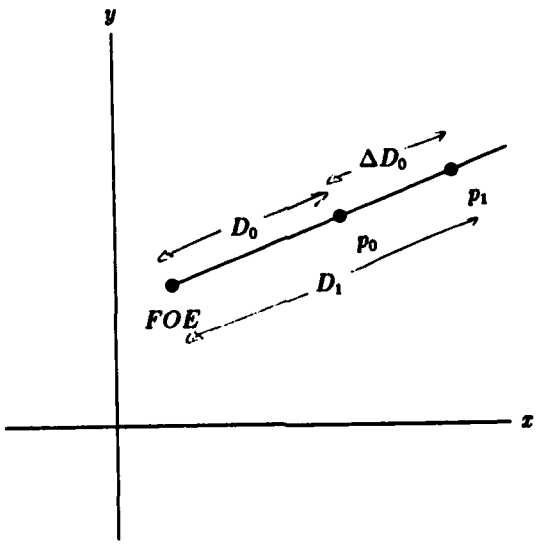


Figure 2: The displacement path

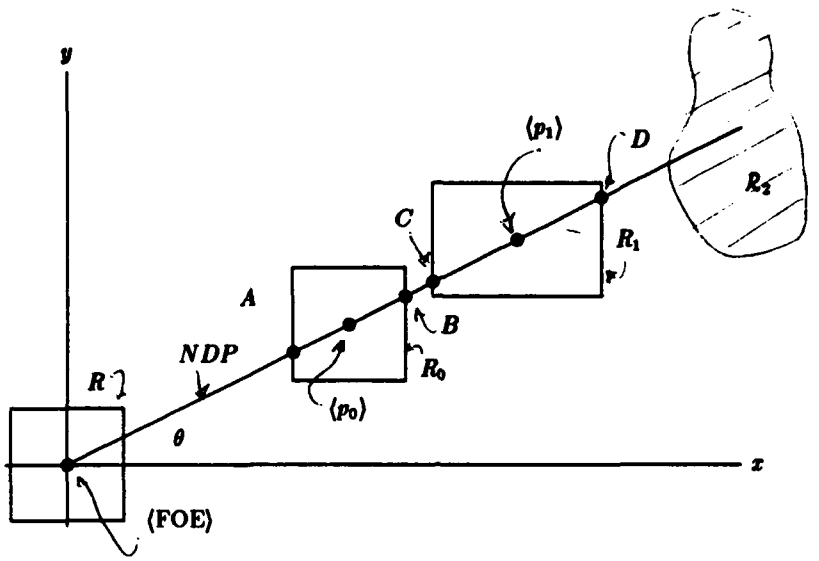


Figure 3: The NDP passes through the vertical sides of all the uncertainty rectangles (Case VVV)

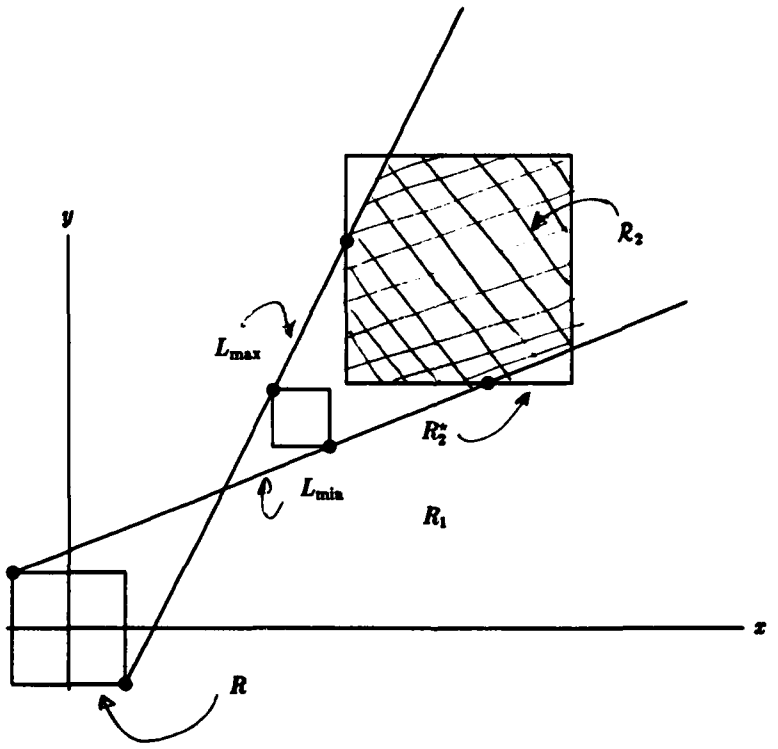


Figure 4: The search region R_2. The shaded region R_2 is the intersection of the search rectangle R_2^* with the "wedge" lying below L_{max} and above L_{min}.

Results of Motion Estimation With More Than Two Frames*

Hormoz Shariat
and
Keith E. Price

Institute for Robotics and Intelligent Systems
Electrical Engineering - Systems
University of Southern California
Los Angeles, CA 90089-0273

Abstract

This paper reports on the results of a general motion estimation system based on using three or more frames. The entire sequence is treated as a whole, thus simplifying the process of estimating the motion parameters, and enabling the computation of the motion parameters in terms of the natural center of motion. We present a brief description of the method, and results for real and synthetic images for the case of 1 point in 5 (and 6) frames.

1. Introduction

Motion analysis has been a problem in computer vision for a long time, with many different approaches and tasks. The problem of estimating motion parameters from a sequence is one of the many tasks in the motion analysis domain. In this paper, we discuss a new method for the estimation of three-dimensional motion parameters given a sequence of views of a moving scene. The application is general motion estimation, but the immediate domain is in an autonomus land vehicle where the vehicle and objects in the scene can both move, but where the vehicle (camera) motion is generally known. We have adopted some concepts from other researchers, but the primary difference is to consider the entire sequence at the same time rather than to consider the sequence as a set of pairs of images. This produces many important constraints that do not exist when only two frames are used. These constraints are used to derive equations which can be used to compute the motion parameters when a few points are matched in a few frames of the sequence.

We first summarize past work in motion estimation through a tabular listing. This is followed by the brief description of our method, with the details given in [1]. Results of applying this method to synthetic and real motion sequences are given.

*This research was supported in part by DARPA contracts DACA76-85-C-0009 and F33615-84-K-1404, order No. 3119 and monitored by the U.S. Army Engineer Topographic Laboratories and the Air Force Wright Aeronautical Laboratories respectively.

Table 1-1 gives an overview of many of the different methods for motion estimation. These are classified according to the number of points required in how many frames (included are methods that use lines or optical flow as input); the type of equations generated (linear, non-linear, polynomial); the kinds of results presented; a short comment on the major strengths and limitations; and references. This table is meant to give an overview of various current methods, and not a complete evaluation of all the methods.

2. Overview of the Multiframe Method

In this section, we introduce our method and its implementation. The actual task of motion estimation is done by the four modules: "General Motion Estimator," "Pure Rotation Estimator," "Pure Translation Estimator," and "Initial Guess Generator." The General Motion Estimator and the Initial Guess Generator are discussed in [1]. The pure rotation and translation cases are treated in [16].

1. Feature Correspondence Extractor - For the details of the methods used for the results reported here see [17] and [18].
2. Motion Classifier - The program picks the category that best describes the behavior of the point on the image plane.
3. Pure Rotation Estimator - This module assumes that the input data represent a pure rotation, and estimates the rotation parameters which best fit the input data. This is discussed more fully in [16].
4. Pure Translation Estimator - This module assumes that the input data represent a pure translation, and estimates a translation vector which best fits the input data. This output can also be used as a quick collision detector. This is discussed more fully in [16].
5. Translation Compensator - This module adjusts the input data to compensate for the computed translation.
6. Rotation Compensator - This module adjusts the input data to compensate for the computed rotation.

No. Points in Frames	Equation Type	Results	Strength	Limitation	Source
OF+Depth	Nonlinear	Synthetic	Solve equations using Hough	Need depth	[2]
1 in 25+	Nonlinear Kalman Filter	Synthetic	Large noise	Complexity many matches	[3]
100+ frames	Linear	Real data	No matches	Much data complex for general	[4]
2 in 4	Nonlinear	Real data	Jointed rigid objects	Parallel projection	[5]
4 in 3	2^{nd} order polynomial	none	Simple equations	no Structure	[6]
2 in 3	2^{nd} order polynomial	none	Simple equations	Parallel projection Needs rotation	[7]
1 in 3	2^{nd} order Polynomial	none	Simple equations	Rotation only	[8]
3 in 3 or 2 in 4 or 1 in 5	Mostly 2^{nd} order polynomials	none none real data	Simple equations no initial guess	Small motions	[9]
4 lines in 3	2^{nd} order polynomial	Synthetic lines	Simple equations	need initial guesses	[10]
6 in 3	Nonlinear	Synthetic	Uses lines	Sensitive to errors	[11]
2 in 3	Mostly Linear	Synthetic	Simple	only Parallel Projection	[12]
5 in 2	2^{nd} Order	Real and Synthetic	Rigidity Constraint	Only 2 frames	[13]
6 in 2	Linear	Synthetic	Rigidity Theoretical analysis	Restricted surfaces	[14]
8 in 2	Linear	Synthetic	Unique solution	Restricted point location	[15]

Table 1-1: Motion Estimation Using Feature Point Matches
Survey of past results

7. Initial Guess Generator - This module automatically generates initial guesses based on an analysis of the formulation.
8. General Motion Estimator - This module uses the Gauss-Newton method along with the guesses provided by the "Initial Guess Generator" to iteratively solve the nonlinear general motion equations.
9. Error and Confidence Evaluator - For each computed parameter we calculate an estimate of the error in the parameter, and a measure of how confident the program is in its estimated error measure.

3. Results

We have applied the method to a number of computer generated test sequences and a few real image sequences. The test data covered the complete range of possible motions from pure translation to pure rotation, and included both small and large motions. Using stochastical techniques, a thorough error, sensitivity, and performance analysis of the method was also performed.

On the average, the program converged to the right answer for most noiseless data within 10 iterations. For noisy data, the answers were reasonable and their accuracy was directly a function of the amount of "Relative Noise" in the input data. "Relative Noise" is defined as:

$$\textit{Relative Noise} = \frac{\textit{The amount of noise in the disparity vector}}{\textit{The length of the disparity vector}}$$

The average time per iteration for the case of 1 feature in 5 frames was 0.556 seconds and for the case of 1 feature in 6 frames was 0.843 seconds on the Symbolics 3640 (with the garbage collector on). Note that these times are short, especially when compared to the time required for feature extraction and matching (or calculating optical flow).

The coordinate system is located at the focal point of the camera. It is oriented such that the plane defined by X- and Y-axes is parallel to the image plane, and the Z-axis is pointing away from the camera.

For synthetic test data, all measurements and calculations are given in terms of multiples (or fractions) of the focal length, which is assumed to be 1. For the real data, the focal length is 50 millimeters and all

measurements and calculations are in millimeters. Note that for the real cases, the dimensions of the images (given in x and y directions) are 36×24 mm. (for the "Turning Car" sequence) and 36×36 mm. for the "Crossing Car" sequence. Also, notice that in *all* cases, the calculated motions are scaled to the camera's focal plane. (This was arbitrarily chosen since the absolute depth information--the scaling--is lost during the imaging process.)

3.1. Synthetic Test Data

It is intended that the results given in this section serve only as *examples* of the behavior of the program under various types of motion. One must study the statistical results reported in [1] and [16] before making any general conclusions about the performance of the program. Synthetic test data was generated to test all portions of the motion program. The three test cases reported here are for general motion only.

1. Case #1 - An "easy" case, where both the translation and rotation are large.
2. Case #2 - A "typical" case.
3. Case #3 - A "worst" case, where both the translation and rotation are very small. This case was chosen to study the source and effect of various sources of errors.

Table 3-1 lists the parameters for each of these cases. Note that in this table (and other tables in this section), the direction of the axis of rotation, which has unit length, is identified with the angles γ and λ which are defined in Figure 3-1. The relative noise in the Table is caused by the (simulated) digitization process on the image plane (assuming a focal length of 1, a 60° field of view and a 512*512 image plane), and represents the *average* noise of the disparity values. In this table, D_{avg} is the average magnitude of disparity vectors given in pixel units.

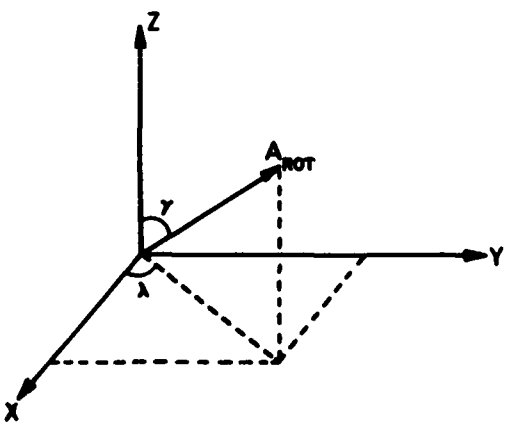


Figure 3-1: Definition of Angles: γ and λ

For each of these 3 cases, the corresponding motion parameters were used to generate an imaginary moving point in space. Then, the image of this moving point on the image plane was sampled to get our test data. This could give either "ideal" (noise free) test data or simulated digital data.

Table 3-2 has the results for both the 5 and 6 frame cases, including the calculation errors and number of iterations needed for convergence. The table also gives the range of error and performance measures generated by the program for each test case.

Figure 3-2 shows the convergence of the best 3 guesses to the solution. Figures 3-3 through 3-5 compare the calculated 3-D motions with the input data for test cases #1, #2, and #3 (for both the 5 or 6 frame case). The tags (arrows) denote the location of the input data (hollow circles) at the time of each frame. The curve represents the calculated 3-D motion after it is interpolated and projected back onto the image plane.

As Table 3-2 shows, even though cases #1 and #2 have reasonable errors, case #3 has large errors. This is due to the fact that the motion (and especially the rotation) is very small, which causes the following problems:

1. T is so small that its information is lost in the noise.
2. Since the axis of rotation is calculated by computing the cross product of two vectors, it is sensitive to the noise in those two vectors. Thus, if the relative noise in each of those two vectors is high, then the noise in the axis of rotation is even higher.
3. Since the disparity vectors on the image plane are small (their average is 4.9 pixels), the relative image digitization error becomes high (for this case it was 14.6% of the average disparity).
4. Because the disparities are small, the matrices used for the Gauss-Newton solution scheme become nearly singular, which cause large errors during their inversion.
5. The round-off errors of the calculations (i.e. the precision of the computer) start having deteriorating effects.

From this table (3-2), we may also see that observing an extra frame (the 6th frame) does improve the results for case #1, but not for cases #2 and #3. This shows that observing more points or more frames does not necessarily improve the results. If the additional data introduces more noise than useful information, then we expect that the results will become more erroneous. As Figure 3-4 shows, observing the 6th frame introduces a disparity vector which is even smaller than the previous

disparities. This introduces more noise in the input data, and hence, increases the error of the calculated parameters.

It must be noted that according to the Table 3-2, the program has accurately assessed its performance. For example, it gave small error measures for test cases #1 and #2 to show the accuracy of the results. However, it gave high error measures along with high confidence numbers for test case #3. These show that the program is very "confident" that the results are erroneous.

Finally, let us note that because of the proximity of the generated initial guesses to the solution, very few iterations were needed for convergence (in cases #1 and #2).

3.2. Real Test Data

In this section two sequences of real test images are treated. In these two cases, because the exact motion of the object is unknown, we can only *intuitively* determine if the answers are correct. Plus we can compare the results for the different points on the object to determine if the results are reasonably consistent. To determine point correspondences, we used the methods described in [18, 17] to segment the image into regions and then match the regions on the basis of their shape, size and position. The segmentation program was run on each frame separately, without any guidance from the previous frames, and the matching was done as a series of pairwise matches. The feature points used for the motion computations are the centers of the regions in each view. These are the pointes plotted on the images in the display of the results. The times are for the implementation on a Symbolics 3645 with garbage collection on (which improves the perfromance substantially) and include some overhead operations, but not the display of the results.

3.2.1. Turning Car Sequence

Figure 3-6 shows 6 frames of a turning car sequence. This sequence was taken by a motor dirven camera at a rate of about 3 frames per second. The car was being driven at a constant speed while turning in a circle. The segmentation produced many regions in all the images, but most of the regions represented stationary objects. The motion classifier determined that most of the sequences of five or six matching regions (a region that can be traced through five or six views of the

	Test Case #1	Test Case #2	Test Case #3
Motion	*Large Tr & Rot*	Typical Tr & Rot	Small Tr & Rot
T	(1, 2, 3)	(2, 2, 2)	(0.01, 0.02, 0.03)
C_{Rot}	(0, -1, 2)	(1, 1, 3)	(5, 5, 10)
γ	30°	45°	60°
λ	30°	45°	60°
θ	45°	20°	5°
R_{Rot}	1	3	1
Rel. Noise	3.1%	0.7%	14.8%
D_{avg} (pixels)	65.7	181.96	4.94

Table 3-1: General Motion Test Cases

	Test Case #1	Test Case #2	Test Case #3
T (Length)	3.25% (2.49%)	3.71% (1.09%)	225.2% (434.2%)
T (Direction)	0.91° (0.63°)	9.87° (10.8°)	121.0° (20.91°)
A_{Rot} (Direction)	11.35° (8.35°)	52.64° (53.1°)	42.27° (118.1°)
θ	0.96° (0.56°)	5.47° (8.28°)	81.47° (114.1°)
C_{Rot}	0.62% (0.44%)	17.94% (21.76%)	9.26% (9.14%)
R_{Rot}	4.5% (1.46%)	43.67% (54.66%)	97.89% (98.51%)
Iterations	4 (3)	3 (2)	430 (200)
Error Measure	10^{-10}-0.2	10^{-8}-2.25	60-124
Confidence Measure	71-94	71-93	70-95

Table 3-2: Error Performance Results of General Motion Cases for 5 Frames (and for 6 frames)

scene) were pure or dominant rotations. Many of the pure rotation sequences are given in the companion paper [16]. In this section we will present the results for two sets of points (see Figure 3-7 where the results are displayed on the fifth frame of the six frame sequence). The sequence labeled "1" corresponds to the passenger windows that was matched in frames 2 through 6, thus the last point (with the pointer) is the location of the window in the frame after the displayed image. Sequence "2" is the driver's side tail light for frames 1 through 5. The results are summarized in Table 3-3.

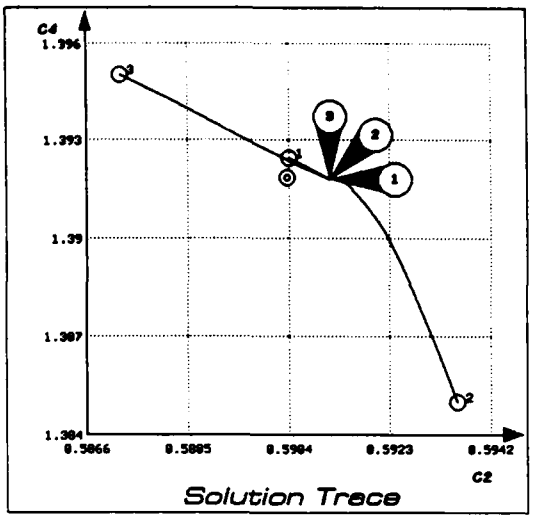
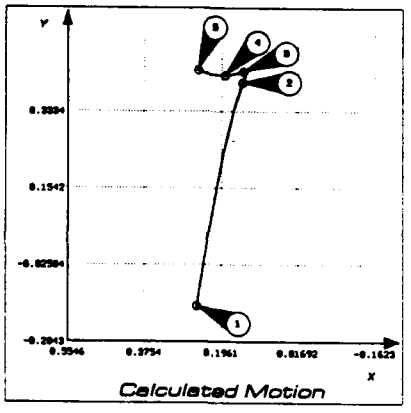
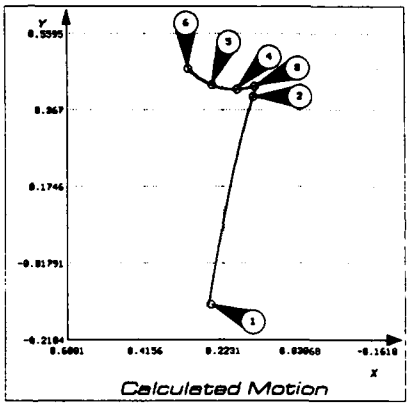


Figure 3-2: Convergence of Top 3 Guesses to a Single Solution

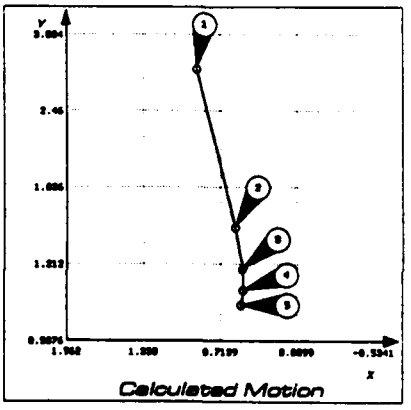


5 Frames Observed

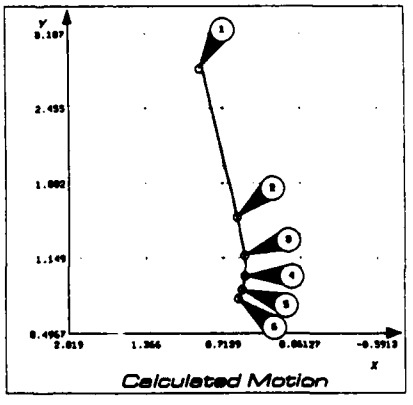


6 Frames Observed

Figure 3-3: Calculated 3-D Motion for Case #1

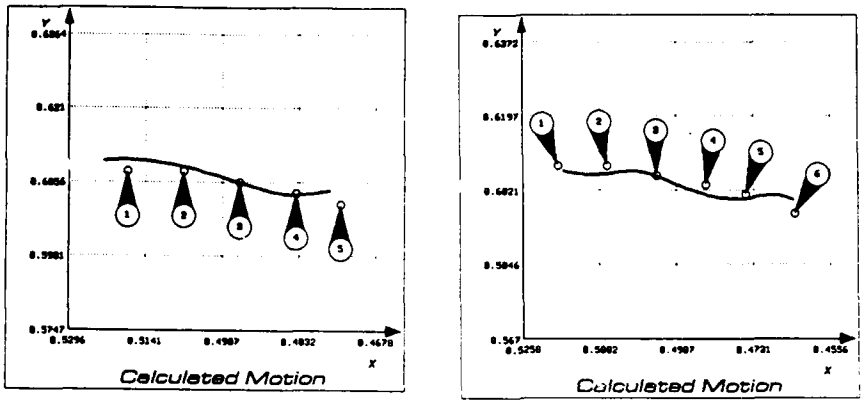


5 Frames Observed



6 Frames Observed

Figure 3-4: Calculated 3-D Motion for Case #2



5 Frames Observed 6 Frames Observed

Figure 3-5: Calculated 3-D Motion for Case #3

(a) 1st, 2nd, and 3rd Frames (b) 4th, 5th, and 6th Frames

Figure 3-6: Six Frames of Turning Car Sequence

	Feature #1	Feature #2
T	(-0.23, -0.60, -10.73)	(-0.33, -0.31, 4.71)
A_{Rot}	(-0.42, -0.86, 0.30)	(0.03 -0.96, 0.28)
θ	69.05°	23.12°
C_{Rot}	(-5.66, 3.46, 50.01)	(-0.60, 0.46, 49.98)
R_{Rot}	0.34	7.08
EM/CM	< .007-13/70-100	11-23/56-100
Time (Sec)	82	5307

Table 3-3: Results for Turning Car Sequence - (General Motion)

The first feature almost fits the translation model (the points appear almost along a straight line); the small radius of rotation shown in Table 3-3 agrees with this. The second feature is very close to a pure rotation, but

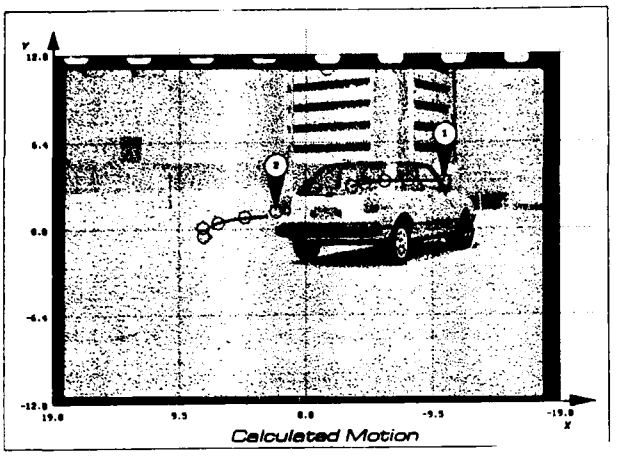


Figure 3-7: Comparison of Input Data (hollow circles) to Calculated Motion (solid curve) for Turning Car

the general motion computation yields a better fit to the data than the rotation computation. The times include approximtely 60 seconds for the calculation of the possible initial guesses and the computation of the residuals for those guesses. The large time for Feature #2 is because many of the initial guesses converged to an invalid set of coefficients (C_2 and C_4 both equal to 0); these values are rejected and the next set of guesses is tried until a valid set of coefficients results.

3.3. Crossing Car Sequence

Figure 3-8 shows 6 frames of the "Crossing Car" sequence. These images were digitized from a video tape, with one image for each field of the television signal (60 frames per second). The first frame for the matched data is frame number 13 with frame number 17 (the fifth in the subsequence) used to display the results. The basic motion is a translation, but because there is some motion of the camera the apparent motion of the truck is not exactly a translation. There are two sets of results, the first is for 5 frames (Figure 3-9) and the second is for six frames (Figure 3-10). The results displayed on frame number 17 have open circles for the centers of the regions which were matched in the input sequence. Tables 3-4 and 3-5 summarize the motion parameters for these two sets of results.

Feature point 1 (Figure 3-9 and Table 3-4) and Feature points 1 and 2 (Figure 3-10 and Table 3-5) are points on the body of the truck. The first few feature point locations are closer together because the entire truck was not visible in these frames. Feature point 2 (5 frame case) is the front wheel. Feature point 3 (5 frames) and Feature points 3 and 4 (6 frames) are the rear wheel of the truck. The motions for the wheels (both the front and rear, for both the five and six frame cases) are consistent with each other; the translations and rotations are very similar. The primary motion is from the translation, with the rotation accounting for the slight up and down motion caused by the camera. Even though the amount of rotation, θ, is large, the radius of rotation is small. Hence the the translation component contains most of the motion information.

The motion for the cab of the truck differs beacuse the location used for the region is its center and the first few views do not include the entire truck body. For this case, the motion computed from six views was much closer to the motion computed for the wheels. Since the Z-component of the translation, T_Z, is calculated from the differences of the spacing of the points on the image plane, small perturbations of these points can translate into large values for T_Z. This causes the inconsistencies for the Z-component in both sets of data (5 frames and 6 frames), while the X- and Y-components are consistent.

The highest error measures (EM) and the lowest confidence measures (CM) are for the computation of the Axis of Rotation (A_{Rot}). This should be expected since A_{Rot} is calculated form the cross product of two vectors and hence is sensitive to noise in each of those vectors. The times include approximately 60 seconds for the computation of the initial guesses for the 5 frame case and 300-400 seconds for the 6 frame case. The extreme time for Feature #1 (Table 3-4) is caused by the rejection of coefficients that are almost zero and the need to try more of the possible guesses until a valid set of final coefficients is obtained.

4. Conclusions

We have presented the results for the computation of three-dimensional motion parameters by using one point in five frames. This method was tested on a set of synthetic data to prove the concept and on a collection of real data to illustrate that it can work well with errors from realistic inputs.

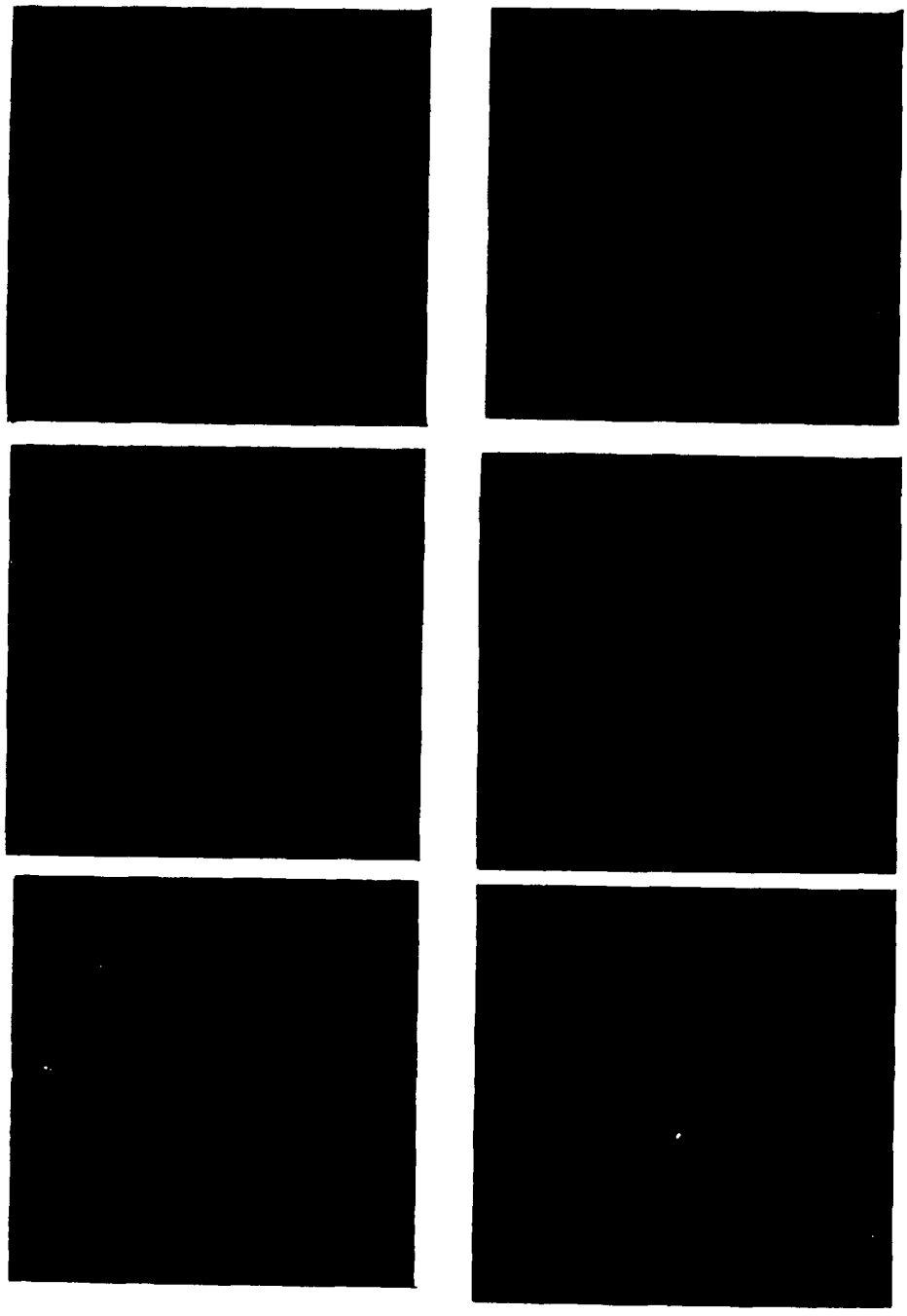
Contributions of this work include:

- Development of an efficient method using the time flow information from three or more frames in a sequence.
- Development of a scheme for automatically generating initial guesses for the solution. The implementation of the three point in three frame case should result in an even better performance with real data.
- Generation of natural parameters of motion, which simplifies comparison of results from one frame to

the next, and accurately predicts the position of the object in subsequent frames.

- Development of a robust software system which has been run on a number of different motion sequences.
- Use of data from existing image matching methods for motion analysis.

This work is not a complete motion analysis system, but is an important part of the final system. There remains much work in combining the motion estimation and prediction capabilities with feature based matching systems to expand the capabilities of both the matching system and the motion estimation system.



(a) 1st, 2nd, and 3rd Frames (b) 4th, 5th, and 6th Frames

Figure 3-8: Six Frames of the Crossing Car Sequence

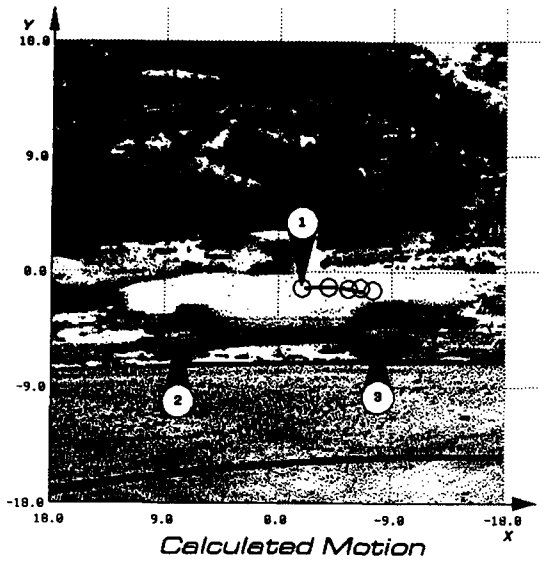


Figure 3-9: Comparison of Input Data (hollow circles) to Calculated Motion (solid curve) (5 Frames) for Crossing Car

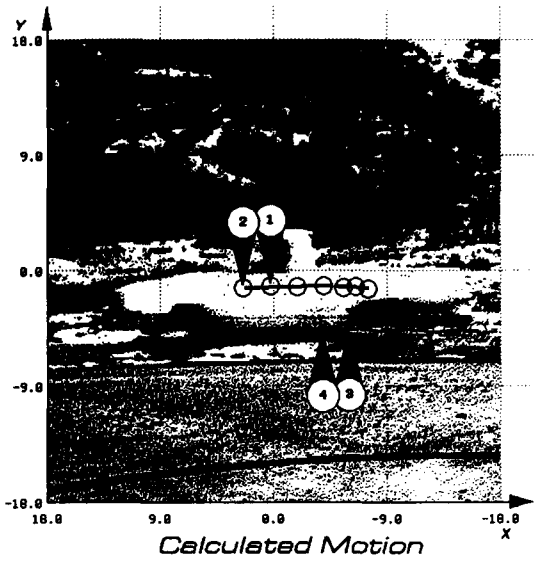


Figure 3-10: Comparison of Input Data (hollow circles) to Calculated Motion (solide curve) (6 Frames) for Crossing Car

	Feature #1	Feature #2	Feature #3
T	(0.83, -0.15, 3.71)	(2.14, -0.01, 0.03)	(2.0, -0.03, 0.45)
A_{Rot}	(0.02, -0.94, -0.34)	(0.17 -0.98, 0.09)	(0.01, 0.84, 0.54)
θ	60.05°	168.6°	156.4°
C_{Rot}	(-4.70, -1.30, 50.19)	(3.67, -4.89, 50.26)	(-12.1, -4.62, 49.91)
R_{Rot}	0.85	0.26	0.11
EM/CM	1.6-44/46-61	10^{-10}/71-87	10^{-9}/71-87
Time (Sec)	9856	193	87

Table 3-4: Results for Crossing Car Sequence - (5 Frames)

	Feature #1	Feature #2
T	(1.92, 0.17, -5.63)	(1.92, 0.09, -4.27)
A_{Rot}	(-0.05, 0.99, -0.15)	(-0.01, 1.0, -0.08)
θ	114.3°	87.37°
C_{Rot}	(-5.53, -1.25, 50.09)	(-3.94, -1.19, 49.76)
R_{Rot}	0.16	0.27
EM/CM	.04-30/75-92	1.6-32/73-90
Time (Sec)	356	317

	Feature #3	Feature #4
T	(1.93, -0.03, 0.67)	(1.95, -0.02, 0.57)
A_{Rot}	(-0.09, 0.73, 0.68)	(-0.11, -0.96, -0.25)
θ	137.0°	171.7°
C_{Rot}	(-12.1, -4.61, 49.95)	(-10.12, -4.63, 50.36)
R_{Rot}	0.08	0.40
EM/CM	.09-39/51-91	1.3-35/55-93
Time (Sec)	622	475

Table 3-5: Results for Crossing Car Sequence - (6 Frames)

References

[1]. H. Shariat and K. Price, "General Motion Estimation with More Than Three Frames," *IEEE Trans. on Pattern Analysis and Machine Intelligence,* Vol. XX, No. YY, Submitted 1986, pp. XX-YY.

[2]. D. H. Ballard and O. A. Kimball, "Rigid Body Motion from Depth and Optical Flow," *Computer Vision, Graphics and Image Processing,* Vol. 22, No. 1, April 1983, pp. 95-115.

[3]. T. J. Broida and R. Chellappa, "Kinematics of a Rigid Object from a Sequence of Noisy Images: A Batch Approach," *Conference on Computer Vision and Pattern Recognition,* IEEE, Miami Beach, FL, June 1986, pp. 176-182.

[4]. R. C. Bolles and H. H. Baker, "Epipolar-Plane Image Analysis: A Technique for Analyzing Motion Sequences," *IEEE Proc. of the 3rd Workshop on Computer Vision: Representation and Control,* Bellaire, MI, October 1985, pp. 168-178.

[5]. J. A. Webb and J. K. Aggarwal, "Structure from Motion of Rigid and Jointed Objects," *Analysis Intelligence,* Vol. 19, No. 1, September 1982, pp. 107-130.

[6]. D. T. Lawton, "Constraint-Based Inference from Image Motion," *Proc. of the 1st Annual National Conference on Artificial Intelligence,* Stanford Univ., August 1980.

[7]. D. D. Hoffman and B. E. Flinchbaugh, "The Interpretation of Biological Motion," *Biological Cybernetics,* Vol. 42, No. 3, 1982, pp. 195-204.

[8]. B. L. Yen and T. S. Huang, "Determining 3-D Motion and Structure of a Rigid Body Using the Spherical Projection," *Computer Graphics and Image Processing,* Vol. 21, 1983, pp. 21-32.

[9]. H. Shariat and K. E. Price, "How to Use More Than Two Frames to Estimate Motion," *Proc. of IEEE Workshop on Motion: Representation and Analysis,* Charlston, SC, May 1986.

[10]. A. Mitiche, S. Seida, and J. Aggarwal, "Line-Based Computation of Structure and Motion Using Angular Invarience," *Workshop on Motion: Representation and Analysis,* IEEE, Kiawah Island, SC, May 1986, pp. 175-180.

[11]. Y. Liu and T. S. Huang, "Estimation of Rigid Body Motion Using Straight Line Correspondences," *Workshop on Motion: Representation and Analysis,* IEEE, Kiawah Island, SC, May 1986, pp. 47-52.

[12]. C.-H. Lee and A. Rosenfeld, "Structure and Motion of a Rigid Object Having Unknown Constant Motion," *Workshop on Motion: Representation and Analysis,* IEEE, Kiawah Island, SC, May 1986, pp. 145-150.

[13]. A. Mitiche, S. Seida, and J. K. Aggarwal, "Determining Position and Displacement in Space from Images," *Proc. of IEEE Conference on Computer Vision and Pattern Recognition,* San Francisco, CA, June 1985, pp. 504-509.

[14]. X. Zhuang and R. Haralick, "Two View Motion Analysis," *Proc. of IEEE Conference on Computer Vision and Pattern Recognition,* San Francisco, CA, June 1985, pp. 686-690.

[15]. R. Y. Tsai and T. S. Huang, "Uniqueness and Estimation of Three-Dimensional Motion Parameters of Rigid Objects with Curved Surfaces," *IEEE Trans. on Pattern Analysis and Machine Intelligence,* Vol. 6, January 1984, pp. 13-27.

[16]. H. Shariat and K. Price, "Motion Estimation Using Multiple Frames Simple Cases: Rotation and Translation," *IEEE Trans. on Pattern Analysis and Machine Intelligence,* Vol. XX, No. YY, Submitted 1986, pp. XX-YY.

[17]. R. Ohlander, K. Price, and D. R. Reddy, "Picture Segmentation Using A Recursive Region Splitting Method," *Computer Graphics and Image Processing,* Vol. 8, 1978, pp. 313-333.

[18]. O. D. Faugeras and K. E. Price, "Semantic Description of Aerial Images Using Stochastic Labeling," *IEEE Trans. on Pattern Analysis and Machine Intelligence,* Vol. 3, No. 6, November 1981, pp. 633-642.

TRACING FINITE MOTIONS WITHOUT CORRESPONDENCE

Ken-ichi Kanatani*
Tsai-Chia Chou

Center for Automation Research
University of Maryland
College Park, MD 20742

ABSTRACT

The 3D motion of an object having planar faces is traced, starting from a known position, from a sequence of 2D perspective projection images *without using any knowledge of point-to-point correspondence*. Computation is based on the shape of the region on the image plane corresponding to a planar face of the object. Given two images, a heuristic guess about the motion is first computed, and one image is transformed according to this estimated motion so that it is positioned close to the other image. Then, the motion that accounts for the remaining small discrepancy is estimated by measuring numerical *features* of the planar regions. The scheme is based on the optical flow due to infinitesimal motion, and estimation is done by solving a set of simultaneous *linear* equations. This process is iterated; after each estimate of the motion, one image is transformed according to the estimated motion so that it is positioned closer and closer to the target image. Various practical issues such as choice of features, constrained motions, face identification, and computation of features without actually transforming the images are discussed. Some numerical examples are also given.

1. INTRODUCTION

Detection of the 3D structure of a moving object from a sequence of 2D images is one of the most important problems in computer vision and is often referred to simply as "structure from motion". The problem takes considerably different forms according to the magnitude of motion that takes place between two successive frames, i.e., "finite motion" or "infinitesimal motion".

Given a point-to-point correspondence between two successive frames, vectors are obtained on the image plane indicating where each point moves from one frame to the next. Let us call these vectors the *displacement field*. If the motion is infinitesimally small, the displacement field is identified with the velocities of image points (the time interval between the two frames is regarded as unit time) and is often termed the *optical flow*.

Analysis of the displacement field is done by solving a set of simultaneous equations which requires rigidity of the object, i.e., constancy of length and angle. Unfortunately, these equations are usually non-linear equations of a complicated form, so that their solution is given only by numerical schemes [23, 25, 29, 30]. The analysis becomes simpler to some extent for optical flows resulting from infinitesimal motion [9, 31], and exact analytical solutions have been obtained in closed forms when the object is a planar surface [18, 19, 24, 28].

The most serious drawback of this type of analysis, whether for the displacement field or for the optical flow, lies in the difficulty of finding the exact point-to-point correspondence. Many techniques of detecting the displacement field or optical flow have been proposed [1, 6 - 9, 11, 12, 21, 26, 30], but they consume considerable computation time. Correspondence cannot be obtained in regions where occlusion takes place. Even if no occlusion is involved, existence of noise or misdetection of correspondence is inevitable. All the above mentioned analyses are known to be very vulnerable to noise (cf. Adiv [2], Tsai and Huang [29]). Therefore, schemes of 3D recovery which do not require correspondence are desired.

There exist methods of computing the optical flow that do not require detection of feature points or point-to-point correspondence when planar regions are identified. Waxman and Wohn [32] observed time varying contour images. Kanatani [13 - 17] computed a set of numerical *features* (which are also called *properties* in Rosenfeld and Kak [26]), measuring characteristics of the shape, such as area, contour length, centroid and moments for a planar region in each frame. However, since these methods are based on optical flow, they can be applied only to infinitesimal motions.

After many attempts, researchers today seem to have agreed that simultaneous detection of both structure and motion from an image sequence based on point-to-point correspondence is extremely difficult and hence impractical. Now, attempts are being made to determine the structure and motion separately. For example, Lin et al. [22] proposed to use stereo to determine the position of the object in each frame and then compute the motions that have taken place from frame to frame. Computation is based on several feature points taken from the object image. According to their scheme, detection of point-to-point correspondence is necessary for stereo matching, but no correspondence is required between image frames taken at different times. Aloimonos and Basu [3] also proposed to use two cameras and feature points taken from the object image. They showed that even the point-to-point correspondence for stereo matching is not necessary if the object is a planar surface.

Another issue is the ambiguity of the interpretation; there exist multiple solutions which correspond to the same displacement field or optical flow. One way to avoid this indeterminacy is to trace the motion starting from a known initial position detected by some other means - range sensor or stereo, for example. This idea combined with the correspondenceless approach was also proposed by Kanatani [13 - 17] for

*Permanent Address: Department of Computer Science, Gunma University, Kiryu, Gunma 376, Japan

a sequence of infinitesimally small motions.

In this paper, we propose a new scheme to trace the motion of an object from a known initial position without using point-to-point correspondence. Furthermore, the motions need not be infinitesimally small. The time interval between successive frames can be arbitrarily long as long as the object of interest is always seen. Here, we assume that a planar face of the object is identified. The motion is computed by measuring numerical "features" characterizing the shape of the planar region.

This type of approach was first tried by Cyganski and Orr [10] under restricted situations. They assumed that the projection is orthographic and that the initial frame gives the image of a planar region parallel to the image plane. They computed various moments of the planar region before and after the motion and reconstructed the "affine" transformation resulting from the motion, so that information about depth could not be obtained. In this paper, on the other hand, we assume perspective projection and the position of the object is arbitrary in the first frame.

First, given two images of successive frames, a heuristic "guess" about the motion is computed, and one image is transformed according to this estimated motion so that it is positioned close to the other image. Then, the motion that accounts for the remaining small discrepancy is estimated by measuring numerical "features" characterizing the shape of the planar region. Estimation is done by solving a set of simultaneous *linear* equations based on the equation of optical flow developed by Kanatani [16, 17]. This process is iterated; after each estimate of the motion, one image is transformed according to the estimated motion so that it is positioned closer and closer to the target image.

The application of our procedure is not limited to motion detection. Evidently, the same procedure can be used to determine the orientation and position of a stationary object in the scene when the shape of the object is known and stored in a data base. Since the shape is known, we can generate, say by a computer graphics technique, the projected image of the object when placed in an appropriate position and orientation. We can locate the object by computing the "motion" between this reference image and the observed image.

Various practical issues which become relevant in implementation are discussed, including the choice of features, constrained motions and face identification. We also discuss a method of computing features without actually transforming images by introducing the "conjugate feature transformation". This method makes it possible to do all the computations on the initially given images alone.

We give some numerical examples. We also test noise sensitivity. Our scheme turns out to work well for a very large motion, and the computation is very robust to noise added to the images and input data.

2. DISPLACEMENT FIELD AND OPTICAL FLOW

2.1 Geometry of Perspective Projection

Consider an xyz-coordinate system fixed in the scene, and regard the xy-plane as the image plane. We regard point $(0,0,-f)$ on the negative side of the z-axis at distance f from the xy-plane as the viewpoint or camera focus, and call f the *focal length*. A point in the scene is projected to the intersection of the xy-plane with the ray connecting the point and the viewpoint (Fig. 1). Hence, a point (X,Y,Z) is projected to point (x,y) on the image plane, where

$$x=\frac{fX}{f+Z}, \qquad y=\frac{fY}{f+Z}. \tag{2.1}$$

This perspective projection reduces to orthographic projection in the limit of $f\to\infty$.

Consider a planar surface S in the scene, and let $z=px+qy+r$ be its equation. We call p, q, r the *surface parameters* of the plane. The 3D position of a point is recovered from its image if the point is known to be on the surface. Namely, point (x,y) is *back-projected* to point (X,Y,Z), where

$$X=\frac{(f+r)x}{f-px-qy}, \qquad Y=\frac{(f+r)y}{f-px-qy}, \qquad Z=\frac{f(px+qy+r)}{f-px-qy}. \tag{2.2}$$

2.2 Finite Motion and Displacement Field

Consider another planar surface S' in the scene. Suppose we observe surface S at time t and surface S' at time t' and we are required to determine the motion in between. In order to specify the rigid motion, choose an arbitrary *reference point* (X_0,Y_0,Z_0) on the surface S. For example, if the surface S is bounded, we may choose an arbitrary point (x_0,y_0) inside the bounded region and backproject it onto the surface S by eqns (2.2). Then, the motion is specified by the translation (a,b,c) of the reference point and the rotation $R=(r_{ij})$ around it (Fig. 2). We call r_{ij}, $i,j=1,2,3$, and a, b, c the *motion parameters*. This motion maps point (X,Y,Z) on the surface to (X',Y',Z'), where

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = \begin{bmatrix} X_0 \\ Y_0 \\ Z_0 \end{bmatrix} + \begin{bmatrix} a \\ b \\ c \end{bmatrix} + R \begin{bmatrix} X-X_0 \\ Y-Y_0 \\ Z-Z_0 \end{bmatrix} \tag{2.3}$$

It is not difficult to prove

Proposition 1. Surface $z=px+qy+r$ is mapped to surface $z=p'x+q'y+r'$ by the motion (2.3), where

$$p'=-\frac{r_{11}p+r_{12}q-r_{13}}{r_{31}p+r_{32}q-r_{33}}, \quad q'=-\frac{r_{21}p+r_{22}q-r_{23}}{r_{31}p+r_{32}q-r_{33}}, \tag{2.4}$$
$$r'=(Z_0+c)-(X_0+a)p'-(Y_0+b)q'.$$

If we substitute eqns (2.2) in the right-hand side of eqn (2.3) and put $x'=fX'/(f+Z')$ and $y'=fY'/(f+Z')$, we obtain

Proposition 2 (Displacement Field). Point (x,y) on the image plane is mapped to point (x',y') by the motion (2.3) according to

$$x'=f\frac{A_{11}x+A_{12}y+A_{13}f}{A_{31}x+A_{32}y+A_{33}f}, \qquad y'=f\frac{A_{21}x+A_{22}y+A_{23}f}{A_{31}x+A_{32}y+A_{33}f}, \tag{2.5}$$

where

$$A_{11}\equiv-\frac{1}{f+r}(p(X_0+a)-(pX_0+f+r)r_{11}-pY_0r_{12}-p(Z_0+f)r_{13}),$$

$$A_{12}\equiv-\frac{1}{f+r}(q(X_0+a)-qX_0r_{11}-(qY_0+f+r)r_{12}-q(Z_0+f)r_{13}),$$

$$A_{13}\equiv\frac{1}{f+r}(X_0+a-r_{11}X_0-r_{12}Y_0-r_{13}(Z_0-r)),$$

$$A_{21}\equiv-\frac{1}{f+r}(p(Y_0+b)-(pX_0+f+r)r_{21}-pY_0r_{22}-p(Z_0+f)r_{23}),$$

$$A_{22}\equiv -\frac{1}{f+r}(q(Y_0+b)-qX_0r_{12}-(qY_0+r+f)r_{22}-q(Z_0+f)r_{23}) \quad (2.6)$$

$$A_{23}\equiv \frac{1}{f+r}(Y_0+b-r_{21}X_0-r_{22}Y_0-r_{23}(Z_0-r)),$$

$$A_{31}\equiv -\frac{1}{f+r}(p(Z_0+c+f)-(pX_0+f+r)r_{31}-pY_0r_{32}-p(Z_0+f)r_{33}),$$

$$A_{32}\equiv -\frac{1}{f+r}(q(Z_0+c+f)-qX_0r_{31}-(qY_0+r+f)r_{32}-q(Z_0+f)r_{33}),$$

$$A_{33}\equiv \frac{1}{f+r}(Z_0+c+f+r-r_{31}X_0-r_{32}Y_0-r_{33}(Z_0-r)-r).$$

Transformation (2.5) is regarded as a 2D *projective transformation*, and is expressed as a linear transformation if *homogeneous coordinates* are used, though the form of eqns (2.5) is more convenient for our purpose. One important consequence is the fact that the matrix $A=(A_{ij})$ induces a *homomorphism* from the transformation group of the form of eqns (2.5) into the group of matrices under multiplication. Namely, if the image undergoes a transformation specified by matrix A followed by another transformation specified by matrix A', the composite transformation is specified by matrix $A''=AA'$. Thus, the inverse transformation of eqns (2.5) is given by replacing matrix A by its inverse A^{-1}. Let us call parameters A_{ij}, $i,j=1,2,3$, the *transformation parameters*.

2.3 Infinitesimal Motion and Optical Flow

As is well known, a 3D rotation is specified by the *rotation axis* (n_1,n_2,n_3), which is taken to be a unit vector, and the *rotation angle* Ω (rad) screwwise around it. The corresponding rotation matrix is given by

$$R=\begin{bmatrix} \cos\Omega+(1-\cos\Omega)n_1^2 & (1-\cos\Omega)n_1n_2-\sin\Omega n_3 & (1-\cos\Omega)n_1n_3+\sin\Omega n_2 \\ (1-\cos\Omega)n_2n_1+\sin\Omega n_3 & \cos\Omega+(1-\cos\Omega)n_2^2 & (1-\cos\Omega)n_2n_3-\sin\Omega n_1 \\ (1-\cos\Omega)n_3n_1-\sin\Omega n_2 & (1-\cos\Omega)n_3n_2+\sin\Omega n_1 & \cos\Omega+(1-\cos\Omega)n_3^2 \end{bmatrix} \quad (2.7)$$

Consider as a special case an infinitesimal motion. If the rotation angle Ω is close to zero, the rotation is close to the identity. Then, matrix R of eqn (2.7) takes the form $R=I+\Delta R$, where I is the unit matrix and

$$\Delta R=\begin{bmatrix} 0 & -\Omega_3 & \Omega_2 \\ \Omega_3 & 0 & -\Omega_1 \\ -\Omega_2 & \Omega_1 & 0 \end{bmatrix}+\cdots \quad (2.8)$$

Here, we put $\Omega_1=\Omega n_1$, $\Omega_2=\Omega n_2$ and $\Omega_3=\Omega n_3$, and $\cdots$ denotes higher order terms in Ω. This infinitesimal rotation is regarded as a rotation around an axis whose orientation is $(\Omega_1,\Omega_2,\Omega_3)$ by angle $\sqrt{\Omega_1^2+\Omega_2^2+\Omega_3^2}$ (rad) screwwise. Thus, a, b, c, Ω_1, Ω_2, Ω_3 are the motion parameters of infinitesimal motion. Then, Propositions 1 and 2 respectively become

Proposition 3. Surface $z=px+qy+r$ is mapped by an infinitesimal motion to surface $z=p'x+q'y+r'$, where $p'=p+\Delta p$, $q'=q+\Delta q$, $r'=r+\Delta r$ and

$$\Delta p=pq\Omega_1-(p^2+1)\Omega_2-q\Omega_3+\cdots,\quad \Delta q=(q^2+1)\Omega_1-pq\Omega_2+p\Omega_3+\cdots,$$
$$\Delta r=-pa-qb+c-(Y_0+q(Z_0-r))\Omega_1+(X_0+p(Z_0-r))\Omega_2+(qX_0-pY_0)\Omega_3+\cdots \quad (2.9)$$

(Here, $\cdots$ denotes higher order terms in a, b, c, Ω_1, Ω_2, Ω_3.)

Proposition 4 (Optical Flow). Point (x,y) on the image plane is mapped by an infinitesimal motion to point (x',y') by $x'=x+\Delta x$, $y'=y+\Delta y$ and

$$\Delta x=u_0+Ax+By+(Ex+Fy)x+\cdots,$$
$$\Delta y=v_0+Cx+Dy+(Ex+Fy)y+\cdots, \quad (2.10)$$

where $\cdots$ denotes higher order terms in a, b, c, Ω_1, Ω_2, Ω_3, and

$$u_0=\frac{f}{f+r}(a-(Z_0-r)\Omega_2+Y_0\Omega_3),\quad v_0=\frac{f}{f+r}(b+(Z_0-r)\Omega_1-X_0\Omega_3).$$

$$A=-\frac{1}{f+r}(pa+c-Y_0\Omega_1+(X_0-p(Z_0+f))\Omega_2+pY_0\Omega_3),$$

$$B=-\frac{1}{f+r}(qa-q(Z_0+f)\Omega_2+(qY_0+f+r)\Omega_3),$$

$$C=-\frac{1}{f+r}(pb+p(Z_0+f)\Omega_1-(pX_0+f+r)\Omega_3), \quad (2.11)$$

$$D=-\frac{1}{f+r}(qb+c-(Y_0-q(Z_0+f))\Omega_1+X_0\Omega_2-qX_0\Omega_3),$$

$$E=\frac{1}{f(f+r)}(pc-pY_0\Omega_1+(pX_0+f+r)\Omega_2),$$

$$F=\frac{1}{f(f+r)}(qc-(qY_0+f+r)\Omega_1+qX_0\Omega_2).$$

The eight parameters u_0, v_0, A, B, C, D, E, F are called the *flow parameters*.

2.4 3D Recovery from Correspondence

If an exact correspondence is given for at least four points on the image plane, we can obtain the transformation parameters of eqns (2.5) or the flow parameters of eqns (2.10) by fitting these equations to the observed correspondence pairs. (For eqns (2.5), only the ratio $A_{11}:A_{12}:A_{13}:A_{21}:A_{22}:A_{23}:A_{31}:A_{32}:A_{33}$ is determined.) Then, the surface parameters p, q, r and and the motion parameters a, b, c, r_{ij} (or Ω_1, Ω_2, Ω_3) are obtained by solving eqns (2.6) or (2.11). However, the solution is in general not unique. For eqns (2.11), a complete solution is available in analytical closed form (cf. Kanatani [18]). If the surface parameters p, q, r are assumed to be known, eqns (2.6) or (2.11) are linear in the motion parameters a, b, c, r_{ij} (or Ω_1, Ω_2, Ω_3), so that they are determined by solving a set of linear equations. However, as was discussed in the previous section, an exact correspondence is difficult to find, and this type of analysis is known to be very sensitive to noise and mismatch of the point-to-point correspondence.

3. MOTION ESTIMATION BY FEATURE DETECTION

3.1 Features of Images

Let $X(x,y)$ represent the image. For example, if the image consists of gray levels, the value of $X(x,y)$ designates the image intensity at point (x,y). If the image consists of colors, $X(x,y)$ may be a vector valued function, corresponding to R, G, B. If the image consists of points and lines, function $X(x,y)$ has delta-function-like singularities. In any case, we define a *feature* of image $X(x,y)$ as a *functional*, i.e., a map $F[X]$ from a set of (appropriately restricted) images $X(x,y)$ to the set R of real numbers. (This is also called a *property* in

Rosenfeld and Kak [25].)

The basic principle of our correspondenceless approach is as follows. We observe an appropriate set of features $F_i[X]$, $i=1,2,3,\cdots$, of image $X(x,y)$ in one frame and image $X'(x,y)$ in the next frame and reconstruct (hopefully) the motion of the object from the numerical values of $F_i[X]$ and $F_i[X']$, $i=1,2,3,\cdots$. Thus, there is no need to know the point-to-point correspondence. It is, however, very difficult to obtain simple equations to determine the motion in terms of the feature values for a general object. Here, for simplicity, we assume that planar parts of the object surface are identified on the image plane and that region-to-region correspondence is known. (This issue is discussed later.)

3.2 Infinitesimal Motion

Let us first consider infinitesimal motion. Let $F[X]$ be the value of a feature of a planar region, i.e., the region corresponding to a planar face of the object, in one frame and $F[X']$ be the value of the same feature of the region in the other frame. If the feature functional $F[\,.\,]$ is smooth (with respect to an appropriate topology), the value of $F[X']$ is also infinitesimally different from $F[X]$, i.e., the difference

$$\Delta F[X] \equiv F[X'] - F[X] \tag{3.1}$$

is infinitesimally small. The planar region under consideration undergoes a small change according to the optical flow of eqns (2.10). The optical flow is linear in the flow parameters u_0, v_0, A, B, C, D, E, F and is zero if these flow parameters are zero. Hence, unless the feature functional $F[\,.\,]$ is pathological, the difference $\Delta F[X]$ is expanded into Taylor series

$$\Delta F[X] = C_{u_0}[X]u_0 + C_{v_0}[X]v_0 + C_A[X]A + C_B[X]B + C_C[X]C + C_D[X]D + C_E[X]E + C_F[X]F + \cdots, \tag{3.2}$$

where ... denotes higher order terms in the flow parameters u_0, v_0, A, B, C, D, E, F.

An important fact is that $C_{u_0}[\,.\,]$, $C_{v_0}[\,.\,]$, $C_A[\,.\,]$, $C_B[\,.\,]$, $C_C[\,.\,]$, $C_D[\,.\,]$, $C_E[\,.\,]$, $C_F[\,.\,]$ are functionals that are derived from the given functional $F[\,.\,]$ *independent of the image* and hence they are *known functionals* given beforehand.

If we substitute eqns (2.11) in eqn (3.2), we obtain

$$\Delta F[X] = C_a[X]a + C_b[X]b + C_c[X]c + C_{\Omega_1}[X]\Omega_1 + C_{\Omega_2}[X]\Omega_2 + C_{\Omega_3}[X]\Omega_3 + (3:3),$$

where

$$C_a[X] \equiv \frac{1}{f+r}(fC_{u_0}[X] - pC_A[X] - qC_B[X]),$$

$$C_b[X] \equiv \frac{1}{f+r}(fC_{v_0}[X] - pC_C[X] - qC_D[X]),$$

$$C_c[X] \equiv -\frac{1}{f+r}((C_A[X] + C_D[X]) - \frac{1}{f}(pC_E[X] + qC_F[X]))$$

$$C_{\Omega_1}[X] \equiv \frac{1}{f+r}(f(Z_0 - r)C_{v_0}[X] + Y_0C_A[X] - p(f+Z_0)C_C[X] + (Y_0 - q(f+Z_0))C_D[X]) - \frac{1}{f}(pY_0C_E[X] + (qY_0 + f + r)C_F[X])), \tag{3.4}$$

$$C_{\Omega_2}[X] \equiv -\frac{1}{f+r}(f(Z_0 - r)C_{u_0}[X] + (X_0 - p(f+Z_0))C_A[X] - q(f+Z_0)C_B[X] + X_0C_D[X] - \frac{1}{f}((pX_0 + f + r)C_E[X] + qX_0C_F[X])).$$

$$C_{\Omega_3}[X] \equiv \frac{1}{f+r}(f(Y_0C_{u_0}[X] - X_0C_{v_0}[X]) - pY_0C_A[X] - (qY_0 + f + r)C_B[X] + (pX_0 + f + r)C_C[X] + qX_0C_D[X]).$$

If the surface parameters p, q, r are known, $C_a[\,.\,]$, $C_b[\,.\,]$, $C_c[\,.\,]$, $C_{\Omega_1}[\,.\,]$, $C_{\Omega_2}[\,.\,]$, $C_{\Omega_3}[\,.\,]$ are *known functionals*, and hence the values of $C_a[X]$, $C_b[X]$, $C_c[X]$, $C_{\Omega_1}[X]$, $C_{\Omega_2}[X]$, $C_{\Omega_3}[X]$ are computed for image $X(x,y)$. Thus, if the difference $\Delta F[X] = F[X'] - F[X]$ is observed from the two images $X(x,y)$ and $X'(x,y)$, eqn (3.3) gives a *linear constraint* on the motion parameters a, b, c, Ω_1, Ω_2, Ω_3 (neglecting higher order terms). This means that if at least six independent feature functionals $F^{(i)}[X]$, $i=1,2,\cdots$, are provided, a set of *linear* equations is available to determine the motion parameters a, b, c, Ω_1, Ω_2, Ω_3 in the form

$$\begin{bmatrix} C_a^{(1)}[X] & C_b^{(1)}[X] & C_c^{(1)}[X] & C_{\Omega_1}^{(1)}[X] & C_{\Omega_2}^{(1)}[X] & C_{\Omega_3}^{(1)}[X] \\ C_a^{(2)}[X] & C_b^{(2)}[X] & C_c^{(2)}[X] & C_{\Omega_1}^{(2)}[X] & C_{\Omega_2}^{(2)}[X] & C_{\Omega_3}^{(2)}[X] \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ \Omega_1 \\ \Omega_2 \\ \Omega_3 \end{bmatrix} = \begin{bmatrix} F^{(1)}[X'] - F^{(1)}[X] \\ F^{(2)}[X'] - F^{(2)}[X] \\ \cdots \\ \cdots \end{bmatrix}, \tag{3.5}$$

or the least square method can be applied to more than six equations.

Thus, given the surface parameters p, q, r for the initial frame, the motion parameters a, b, c, Ω_1, Ω_2, Ω_3 of the motion between the initial and the next frames are obtained without using point-to-point correspondence. Then, the surface parameters p', q', r' in the next frame are approximately determined by eqns (2.9). This process can be repeated, and the motion of the surface is traced if we start from a known initial position of the surface. This is a special case of the principle proposed by Kanatani [15,16]. He also studied in detail various types of possible features.

3.3 Finite Motion

The method described above is based on optical flow and hence is a first order approximation with respect to the motion parameters. Therefore, errors accumulate unless the time interval between successive frames is very short. On the other hand, if the time interval is too short, the differences $\Delta F^{(i)}[X]$ are very small and are likely to be buried in numerical errors. In the following, we consider a scheme applicable to finite motions.

The basic principle we adopt is to use the method described above to obtain an initial approximation and to transform the image according to the estimated motion. This process is repeated until the two images sufficiently overlap. Let p, q, r be the initially known surface parameters, and let a, b, c, Ω_1, Ω_2, Ω_3 be the estimated motion parameters. Let $\boldsymbol{R}$ be the rotation matrix computed by eqn (2.7) by putting $\Omega = \sqrt{\Omega_1^2 + \Omega_2^2 + \Omega_3^2}$, $n_1 = \Omega_1/\Omega$, $n_2 = \Omega_2/\Omega$, $n_3 = \Omega_3/\Omega$. If the surface undergoes the motion of eqn (2.3), the new surface parameters $\tilde{p}$, $\tilde{q}$, $\tilde{r}$ are given by eqns (2.4). The image is transformed according to the displacement field of eqns (2.5).

Suppose image $X(x,y)$ is transformed by the displace-

ment field of eqns (2.5) into image $\tilde{X}(x,y)$. Since we know the surface parameters $\tilde{p}$, $\tilde{q}$, $\tilde{r}$, we can repeat the same process by measuring the features of image $\tilde{X}(x,y)$. (There exists a method to compute these features from the original iamge $X(x,y)$ without actually doing image transformation; cf. Appendix A.) At this stage, we compute the motion with respect to the new reference point $(\tilde{X}_0,\tilde{Y}_0,\tilde{Z}_0)$, where

$$\tilde{X}_0=X_0+a, \quad \tilde{Y}_0=Y_0+b, \quad \tilde{Z}_0=Z_0+c. \tag{3.6}$$

Thus, the new motion parameters $\tilde{a}$, $\tilde{b}$, $\tilde{c}$, $\tilde{\Omega}_1$, $\tilde{\Omega}_2$, $\tilde{\Omega}_3$ of the next step are given by solving

$$\begin{bmatrix} C_a^{(1)}[\tilde{X}] & C_b^{(1)}[\tilde{X}] & C_c^{(1)}[\tilde{X}] & C_{\Omega_1}^{(1)}[\tilde{X}] & C_{\Omega_2}^{(1)}[\tilde{X}] & C_{\Omega_3}^{(1)}[\tilde{X}] \\ C_a^{(2)}[\tilde{X}] & C_b^{(2)}[\tilde{X}] & C_c^{(2)}[\tilde{X}] & C_{\Omega_1}^{(2)}[\tilde{X}] & C_{\Omega_2}^{(2)}[\tilde{X}] & C_{\Omega_3}^{(2)}[\tilde{X}] \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \end{bmatrix} \begin{bmatrix} \tilde{a} \\ \tilde{b} \\ \tilde{c} \\ \tilde{\Omega}_1 \\ \tilde{\Omega}_2 \\ \tilde{\Omega}_3 \end{bmatrix}$$

$$= \begin{bmatrix} F^{(1)}[X']-F^{(1)}[\tilde{X}] \\ F^{(2)}[X']-F^{(2)}[\tilde{X}] \\ \cdots \\ \cdots \end{bmatrix} \tag{3.7}$$

Now, we obtain the rotation matrix $\tilde{\boldsymbol{R}}$ by eqn (2.7) by putting $\tilde{\Omega}=\sqrt{\tilde{\Omega}_1{}^2+\tilde{\Omega}_2{}^2+\tilde{\Omega}_3{}^2}$ and substituting $\Omega=\tilde{\Omega}$, $n_1=\tilde{\Omega}_1/\tilde{\Omega}$, $n_2=\tilde{\Omega}_1/\tilde{\Omega}$, $n_3=\tilde{\Omega}_3/\tilde{\Omega}$ in it. The composition of two motions specified by motion parameters r_{ij}, a, b, c and $\tilde{r}_{ij}$, $\tilde{a}$, $\tilde{b}$, $\tilde{c}$ is given as follows:

Proposition 5 (Rule of Composition). The composition of a motion specified by motion parameters a, b, c, $\boldsymbol{R}=(r_{ij})$ with respect to reference point (X_0,Y_0,Z_0) followed by a motion specified by motion parameters $\tilde{a}$, $\tilde{b}$, $\tilde{c}$, $\tilde{\boldsymbol{R}}=(\tilde{r}_{ij})$, with respect to reference point $(\tilde{X}_0,\tilde{Y}_0,\tilde{Z}_0)$ is equivalent to a motion specified by motion parameters a', b', c', $\boldsymbol{R}'=(r_{ij}')$, with respect to reference point (X_0,Y_0,Z_0) defined as follows (Fig. 3):

$$a'=a+\tilde{a}, \quad b'=b+\tilde{b}, \quad c'=c+\tilde{c}, \quad \boldsymbol{R}'=\tilde{\boldsymbol{R}}\boldsymbol{R}. \tag{3.8}$$

Using r_{ij}', a', b', c' as improved estimates of the motion parameters, we repeat the whole procedure until all features $F^{(i)}[\tilde{X}]$ become sufficiently close to $F^{(i)}[X']$ of the next frame.

4. FEATURES OF PLANAR REGIONS

4.1 Consistency of Features

A good choice of features is essential to the present scheme. It is crucial that the features be *consistent* in the sense that they must faithfully respond to image distortions due to 3D motion alone and should not be affected by other factors such as reflectance changes. Originally, characterization of an image by "features" (or "properties") was proposed for gray-level images (Amari [4, 5], Rosenfeld and Kak [27], cf. Appendix B). For gray-level images, however, we must impose the consistency requirement that the gray levels are inherent to the object itself and that they are not affected by viewing angle and reflectance changes. This assumption is sometimes satisfied, e.g., when chromaticity is measured. However, in most cases, the absolute values of gray levels do not necessarily reflect only the physical properties of the object, but are affected by numerous factors such as lighting, camera characteristics and digital image processing. Hence, it is desirable to use as features more stable characteristics such as marked feature points, line segments, edges and contours.

4.2 Dot Features

If we can identify specific dots in the region S of the image plane under consideration, we can define a feature as the summation

$$F[X]=\sum_{P_i\in S} m(x_i,y_i), \tag{4.1}$$

of the values of an arbitrary *weight function* $m(x,y)$ over the position $P(x_i,y_i)$ of the dots. This is possible, for example, if the object surface is textured with dots which are clearly visible and identifiable in each frame. However, a one-to-one correspondence between the dots is not necessary. This type of feature was used by Kanatani and Chou [20] in the "shape from texture" problem. Another possibility is to use characteristic points of contours. If the planar region in question is a polygonal face of an object, the corner points can be used as stable feature points.

If each feature point (x_i,y_i) is displaced by $(\Delta x_i,\Delta y_i)$, the value of $F[X]$ changes by

$$\Delta F[X]=\sum_{P_i\in S}\left[\frac{\partial m}{\partial x}(x_i,y_i)\Delta x_i+\frac{\partial m}{\partial y}(x_i,y_i)\Delta y_i+\cdots\right], \tag{4.2}$$

where ... denotes higher order terms in Δx, Δy. Substituting the equations of optical flow (2.10), we find the functionals $C_{u_0}[X]$, $C_{v_0}[X]$, $C_A[X]$, $C_B[X]$, $C_C[X]$, $C_D[X]$, $C_E[X]$, $C_F[X]$ in the form

$$C_{u_0}[X]=\sum_{P_i\in S}\frac{\partial m}{\partial x}(x_i,y_i), \quad C_{v_0}[X]=\sum_{P_i\in S}\frac{\partial m}{\partial y}(x_i,y_i),$$

$$C_A[X]=\sum_{P_i\in S}x_i\frac{\partial m}{\partial x}(x_i,y_i), \quad C_B[X]=\sum_{P_i\in S}y_i\frac{\partial m}{\partial x}(x_i,y_i),$$

$$C_C[X]=\sum_{P_i\in S}x_i\frac{\partial m}{\partial y}(x_i,y_i), \quad C_D[X]=\sum_{P_i\in S}y_i\frac{\partial m}{\partial y}(x_i,y_i), \tag{4.3}$$

$$C_E[X]=\sum_{P_i\in S}\left[x_i{}^2\frac{\partial m}{\partial x}(x_i,y_i)+x_iy_i\frac{\partial m}{\partial y}(x_i,y_i)\right],$$

$$C_F[X]=\sum_{P_i\in S}\left[x_iy_i\frac{\partial m}{\partial x}(x_i,y_i)+y_i{}^2\frac{\partial m}{\partial y}(x_i,y_i)\right].$$

Thus, if we take six independent functions $m_i(x,y)$, $i=1,\ldots,6$, we obtain the corresponding features $F^{(i)}[X]$ in the form of eqn (4.1), and $C_a^{(i)}[X]$, $C_b^{(i)}[X]$, $C_c^{(i)}[X]$, $C_{\Omega_1}^{(i)}[X]$, $C_{\Omega_2}^{(i)}[X]$, $C_{\Omega_3}^{(i)}[X]$, $i=1,\ldots,6$, are given by eqns (3.4). Hence, the motion can be recovered by the procedure described in the previous section.

4.3 Line Features

If we can identify specific line segments in the planar region S under consideration, we can define a feature functional as the sum of the line integrals

$$F[X]=\sum_{L_i\subset S}\int_{L_i}m(x,y)ds \tag{4.4}$$

of an arbitrary weight function $m(x,y)$ along each individual line segment L_i in the planar region S. Again, this type of functional can be applied if the surface is textured by clearly

visible and identifiable line segments (cf. Kanatani and Chou [20]), and no knowledge of the correspondence between line segments is necessary. Alternatively, we can use a single bounded contour in the region as a stable feature (cf. Waxman and Wohn [32], Kanatani [15]).

It follows from elementary calculus that if each point (x,y) on each line segment L_i is displaced by $(\Delta x,\Delta y)$, the value of $F[X]$ changes by

$$\Delta F[X]=\sum_{L_i\subset S}\int_{L_i}[\frac{\partial m}{\partial x}\Delta x+\frac{\partial m}{\partial y}\Delta y$$

$$+(n_1^2\frac{\partial\Delta x}{\partial x}+n_1n_2(\frac{\partial\Delta x}{\partial y}+\frac{\partial\Delta y}{\partial x})+n_2^2\frac{\partial\Delta y}{\partial y})m]ds+\cdots \quad (4.5)$$

where s and (n_1,n_2) are the arc length and unit tangent vector respectively along the line segment L_i. Here ... denotes higher order terms in Δx, Δy and their derivatives. Substituting the equations of optical flow (2.10), we find the functionals $C_{u_0}[X]$, $C_{v_0}[X]$, $C_A[X]$, $C_B[X]$, $C_C[X]$, $C_D[X]$, $C_E[X]$, $C_F[X]$ in the form

$$C_{u_0}[X]=\sum_{L_i\subset S}\int_{L_i}\frac{\partial m}{\partial x}ds,\quad C_{v_0}[X]=\sum_{L_i\subset S}\int_{L_i}\frac{\partial m}{\partial y}ds,$$

$$C_A[X]=\sum_{L_i\subset S}\int_{L_i}[x\frac{\partial m}{\partial x}+n_1^2m]ds,\quad C_B[X]=\sum_{L_i\subset S}\int_{L_i}[y\frac{\partial m}{\partial x}+n_1n_2m]ds,$$

$$C_C[X]=\sum_{L_i\subset S}\int_{L_i}[x\frac{\partial m}{\partial y}+n_1n_2m]ds,\quad C_D[X]=\sum_{L_i\subset S}\int_{L_i}[y\frac{\partial m}{\partial y}+n_2^2m]ds \quad (4.6)$$

$$C_E[X]=\sum_{L_i\subset S}\int_{L_i}[x^2\frac{\partial m}{\partial x}+xy\frac{\partial m}{\partial y}+(2xn_1^2+yn_1n_2+xn_2^2)m]ds,$$

$$C_F[X]=\sum_{L_i\subset S}\int_{L_i}[xy\frac{\partial m}{\partial x}+y^2\frac{\partial m}{\partial y}+(yn_1^2+xn_1n_2+2yn_2^2)m]ds,$$

Thus, if we take six independent functions $m_i(x,y)$, $i=1,\ldots,6$, we obtain the corresponding features $F^{(i)}[X]$ in the form of eqn (4.4), and $C_a^{(i)}[X]$, $C_b^{(i)}[X]$, $C_c^{(i)}[X]$, $C_{\Omega_1}^{(i)}[X]$, $C_{\Omega_2}^{(i)}[X]$, $C_{\Omega_3}^{(i)}[X]$, $i=1,\ldots,6$, are given by eqns (3.4). Hence, the motion is recovered by the procedure described in the previous section.

4.4 AREA FEATURES

If a bounding contour of a planar face is observed, we can also define a feature functional as the area integral

$$F[X]=\int\int_S m(x,y)dxdy, \quad (4.7)$$

of an arbitrary weight function $m(x,y)$ over the region S (cf. Kanatani [15]). The area integral can always be converted into a line integral along the contour C of region S by *Green's theorem*. Namely, we can always find two functions $P(x,y)$, $Q(x,y)$ such that

$$m(x,y)=\frac{\partial Q}{\partial x}-\frac{\partial P}{\partial y}, \quad (4.8)$$

and we obtain

$$\int\int_S m(x,y)dxdy=\int_C[P(x,y)n_1+Q(x,y)n_2]ds, \quad (4.9)$$

where s and (n_1,n_2) are the arc length and unit tangent vector respectively along the contour C counterclockwise.

If each point (x,y) on the image plane undergoes a displacement by $(\Delta x,\Delta y)$, the difference $\Delta F[X]$ of the feature value is expressed either as an area integral over the region S or as a line integral along its contour C in the form

$$\Delta F[X]=\int\int_S[\frac{\partial m}{\partial x}\Delta x+\frac{\partial m}{\partial y}\Delta y+(\frac{\partial\Delta x}{\partial x}+\frac{\partial\Delta y}{\partial y})m]dxdy+\cdots,$$

$$=\int_C(n_2\Delta x-n_1\Delta y)mds+\cdots, \quad (4.10)$$

where ... again denotes higher order terms in Δx, Δy and their derivatives. Substituting the equation of optical flow (2.10), we find

$$C_{u_0}[X]=\int\int_S\frac{\partial m}{\partial x}dxdy=\int_C n_2mds,\ C_{v_0}[X]=\int\int_S\frac{\partial m}{\partial y}dxdy=-\int_C n_1mds,$$

$$C_A[X]=\int\int_S[m+x\frac{\partial m}{\partial x}]dxdy=\int_C xn_2mds,$$

$$C_B[X]=\int\int_S y\frac{\partial m}{\partial x}dxdy=\int_C yn_2mds,$$

$$C_C[X]=\int\int_S x\frac{\partial m}{\partial y}dxdy=-\int_C xn_1mds, \quad (4.11)$$

$$C_D[X]=\int\int_S[m+y\frac{\partial m}{\partial y}]dxdy=-\int_C yn_1mds,$$

$$C_E[X]=\int\int_S[3xm+x^2\frac{\partial m}{\partial x}+xy\frac{\partial m}{\partial y}]dxdy=\int_C(x^2n_2-xyn_1)mds,$$

$$C_F[X]=\int\int_S[3ym+xy\frac{\partial m}{\partial x}+y^2\frac{\partial m}{\partial y}]dxdy=\int_C(xyn_2-y^2n_1)mds.$$

Thus, if we take six independent functions $m_i(x,y)$, $i=1,\ldots,6$, we obtain the corresponding features $F^{(i)}[X]$ in the form of eqn (4.7), and $C_a^{(i)}[X]$, $C_b^{(i)}[X]$, $C_c^{(i)}[X]$, $C_{\Omega_1}^{(i)}[X]$, $C_{\Omega_2}^{(i)}[X]$, $C_{\Omega_3}^{(i)}[X]$, $i=1,\ldots,6$, are obtained by eqns (3.4). Hence, the motion is recovered by the procedure described in the previous section.

5. NUMERICAL EXAMPLES

5.1 Algorithm with Initial Adjustment

In the following, we consider a polygonal region S which is supposed to be the image of a face of an object in the scene. As features, we consider (i) summations with respect to the corner points of the contour C in the form of eqn (4.1), (ii) line integrals along the contour C in the form of eqn (4.4), and (iii) area integrals over the region S in the form of eqn (4.7) converted into line integrals by Green's theorem, eqn (4.9).

Let the region S in the first frame be a polygon with corner points (x_i,y_i), $i=1,\ldots,N$, and the region S' in the second frame be a polygon with corner points (x_i',y_i'), $i=1,\ldots,N$. We compute the arithmetic average (x_0,y_0) of the corner points in the first frame and define the reference point (X_0,Y_0,Z_0) as its backprojection by eqns (2.2). Similarly, we compute (x_0',y_0') for the next frame. We define the "sizes" of the two regions S, S' on the image plane by

$$l_0=\max_i\sqrt{(x_i-x_0)^2+(y_i-y_0)^2},\quad l_0'=\max_i\sqrt{(x_i'-x_0')^2+(y_i'-y_0')^2}. \quad (5.1)$$

We then define the "estimate of the object size" by

$$L_0=(Z_0/f+1)l_0. \quad (5.2)$$

From this size L_0 and the size l_0' of the region S' in the next frame, we can estimate the position of the reference point (X_0',Y_0',Z_0') in the next frame as follows:

$$Z_0'=(L_0/l_0'-1)f, \quad X_0'=(Z_0'/f+1)x_0', \quad Y_0'=(Z_0'/f+1)y_0' \quad (5.3)$$

(Fig. 4). We use the following motion as the initial adjustment:

$$a=X_0'-X_0, \quad b=Y_0'-Y_0, \quad c=Z_0'-Z_0, \quad \Omega_1=\Omega_2=\Omega_3=0. \quad (5.4)$$

From the second iteration, we use the scheme described in Section 3. As the weight functions, we use

$$m_1(x,y)=1/l, \quad m_2(x,y)=(x-x_0)/l, \quad m_3(x,y)=(y-y_0)/l,$$

$$m_4(x,y)=(x-x_0)^2/l^2, \quad m_5(x,y)=(x-x_0)(y-y_0)/l^2, \quad m_6(x,y)=(y-y_0)^2/l^2,$$

$$m_7(x,y)=(x-x_0)^3/l^3, \quad m_8(x,y)=(x-x_0)^2(y-y_0)/l^3, \quad (5.5)$$

$$m_9(x,y)=(x-x_0)(y-y_0)^2/l^3, \quad m_{10}(x,y)=(y-y_0)^3/l^3,$$

where l is another measure of the size of the region S defined by

$$l \equiv \max_i \{|x_i-x_0|, |y_i-y_0|\} \quad (5.6)$$

Since we obtain ten equations for six unknowns in the form of eqns (3.5) and (3.8), we use the least square method to solve them. Since the computation is iterative, the weight functions $m_i(x,y)$, $i=1, \ldots, 10$ of eqns (5.5) and the size l of (5.6) are revised at each iteration step. (We use size l of eqn (5.6) instead of size l_0 of eqns (5.1) simply because size l is computed faster than size l_0.) In the following, we use a scale of length such that $f=1$. The length of $f/2$ is indicated in each figure.

5.2 Simultaneous Computation of Translation and Rotation

Fig. 5 shows an initial region S and the corresponding region S' after motion. The surface parameters (p,q,r) of S and S' are (–0.408, 0.408, 1.000) and (–0.875, 0.399, 2.093) respectively. After the initial adjustment described by eqn (5.4), the estimate of the surface parameters is (–0.408, 0.408, 1.972). (The p, q components are unchanged because the surface is only translated.) The subsequent steps are as follows:

	corner features	contour features	area features
1	(–0.900, 0.219, 2.746)	(–0.953, 0.361, 2.299)	(–1.101, 0.356, 2.390)
2	(–0.474, 0.165, 1.934)	(–0.839, 0.395, 2.143)	(–0.823, 0.368, 2.080)
3	(–1.383, 0.571, 2.265)	(–0.877, 0.399, 2.093)	(–0.878, 0.405, 2.111)
4	(–0.929, 0.363, 2.129)	(–0.875, 0.399, 2.093)	(–0.877, 0.399, 2.087)
5	(–0.864, 0.390, 2.045)	(–0.875, 0.399, 2.093)	(–0.875, 0.399, 2.093)
...	.....	.....	.....
10	(–0.875, 0.400, 2.093)	(–0.875, 0.399, 2.093)	(–0.875, 0.399, 2.093)

Convergence is very fast for contour and area features; only four or five iterations are sufficient. Here and in the following, we only show the surface parameters p, q, r, but they completely determine the 3D position of the surface, since there exists a one-to-one correspondence between a point on the image plane and a point in the scene by eqns (2.2).

The iteration process does not always converge. From experiments, we can conclude that the depth, i.e., the distance of the surface from the image plane, must be small in the second frame. Also, the range of the values that the motion parameters can take is limited. Hence, the applicability of the approach is somewhat restricted if the six degrees of freedom of the motion are to be determined simultaneously. However, the algorithm works very well when the object is near the camera and the motion is not too large.

5.3 Alternate Computation of Translations and Rotations

In the above, we tried to determine the six motion parameters at the same time. It is expected, in general, that the computation becomes more stable and robust as the number of unknowns becomes smaller. Here, noting that translations and rotations are very different types of motion, we first apply the "translational scheme", which is the same as the above except that the rotation is set to zero from the beginning. After computing the estimate of the translation, the image is moved accordingly. Next, we apply the "rotational scheme", which is again the same as the above except that the translation is set to zero from the beginning. After computing the estimate of the rotation, the image is moved accordingly. This process is repeated, applying the translational scheme and the rotational scheme alternately.

Fig. 6 shows an initial region S and the corresponding region S' after motion. The surface parameters (p,q,r) of S and S' are (–0.408, 0.408, 10.00) and (–0.694, 0.382, 22.08) respectively. After the initial adjustment described by eqn (5.4), the estimate of the surface parameters is (–0.408, 0.408, 18.89). The following shows intermediate results after each stage, where one stage consists of the translational scheme iterated six times and the rotational scheme iterated six times.

	corner features	contour features	area features
1	(–0.691, 0.379, 22.08)	(–0.827, 0.443, 22.50)	(–0.739, 0.438, 21.52)
2	(–0.691, 0.377, 22.15)	(–0.779, 0.407, 22.57)	(–0.711, 0.436, 21.29)
3	(–0.692, 0.379, 22.12)	(–0.750, 0.387, 22.59)	(–0.690, 0.434, 21.12)
4	(–0.693, 0.380, 22.10)	(–0.733, 0.375, 22.60)	(–0.676, 0.432, 21.01)
5	(–0.694, 0.381, 22.09)	(–0.723, 0.370, 22.58)	(–0.667, 0.431, 20.95)
...	.....	.....	.....
10	(–0.694, 0.382, 22.08)	(–0.706, 0.368, 22.43)	(–0.654, 0.425, 20.92)

Convergence is very fast for dot features but is slow for contour and area features. However, this method has a far wider range of applicability compared with the simultaneous computation of translation and rotation. The iteration converges for a very wide range of positions, orientations and motions.

5.4 Horizontally Constrained Motion

In many practical situations, motion takes place on a horizontal plane: cars moving on flat ground, objects displaced on a flat tabletop, etc. Let $e_1=(\xi_1,\xi_2,\xi_3)$, $e_2=(\eta_1,\eta_2,\eta_3)$ be two mutually orthogonal unit vectors defining a horizontal plane in reference to our camera-based xyz-coordinate system, and let $n=(n_1,n_2,n_3)$ be the unit vector normal to that plane (Fig. 7). In other words, the camera optical axis need not be horizontal. Then, a translation has the form $\alpha e_1+\beta e_2$ and a rotation is specified by rotation angle Ω around axis n at a prescribed reference point (X_0,Y_0,Z_0). Substituting

$$a=\alpha\xi_1+\beta\eta_1, \quad b=\alpha\xi_2+\beta\eta_2, \quad c=\alpha\xi_3+\beta\eta_3,$$
$$\Omega_1=\Omega n_1, \quad \Omega_2=\Omega n_2, \quad \Omega_3=\Omega n_3 \quad (5.7)$$

in eqn (3.3), we obtain

$$\Delta F[X]=C_\alpha[X]\alpha+C_\beta[X]\beta+C_\Omega[X]\Omega+\ldots, \quad (5.8)$$

where

$$C_\alpha[X]\equiv\xi_1C_a[X]+\xi_2C_b[X]+\xi_3C_c[X],$$
$$C_\beta[X]\equiv\eta_1C_a[X]+\eta_2C_b[X]+\eta_3C_c[X], \quad (5.9)$$
$$C_\Omega[X]\equiv n_1C_{\Omega_1}[X]+n_2C_{\Omega_2}[X]+n_3C_{\Omega_3}[X].$$

Fig. 8 shows an initial region S and corresponding region S' after motion, where e_1=(1,0,0), e_2=(0,0,1), n=(0,1,0) (i.e., the camera optical axis is horizontal). The surface parameters (p,q,r) of S and S' are (–0.408, 0.408, 10.0) and (–0.039, 0.378, 21.1) respectively. After the initial adjustment described by eqn (5.4), the estimate of the surface parameters is (–0.408, 0.408, 23.1). The subsequent steps are as follows:

	corner features	contour features	area features
1	(–0.152, 0.382, 22.1)	(–0.338, 0.399, 23.7)	(–0.388, 0.405, 24.2)
2	(–0.354, 0.401, 23.5)	(–0.087, 0.379, 21.8)	(–0.239, 0.389, 23.1)
3	(0.021, 0.378, 20.7)	(–0.032, 0.378, 21.1)	(0.066, 0.379, 20.3)
4	(–0.107, 0.380, 21.6)	(–0.049, 0.378, 21.2)	(–0.026, 0.378, 21.0)
5	(–0.049, 0.378, 21.2)	(–0.038, 0.378, 21.1)	(–0.039, 0.378, 21.1)
	.....	.....	.....
10	(–0.039, 0.378, 21.1)	(–0.039, 0.378, 21.1)	(–0.039, 0.378, 21.1)

Convergence is very fast, and the applicable range of positions, orientations and motions is larger than for unconstrained motion.

We can again use the scheme of alternate translations and rotations; the translational scheme determines α and β, and the rotational scheme determines Ω. Fig. 9 shows an initial region S and the corresponding region S' after motion, where e_1=(1,0,0), e_2=(0,0,1), n=(0,1,0) (i.e., the camera optical axis is horizontal). The surface parameters (p,q,r) of S and S' are (–0.408, 0.408, 10.0) and (0.137, 0.381, 24.1) respectively. After the initial adjustment described by eqn (5.4), the estimate of the surface parameters is (–0.408, 0.408, 31.7). The following shows intermediate results after each stage, where again one stage consists of the translational scheme iterated six times and the rotational scheme iterated six times.

	corner features	contour features	area features
1	(0.050, 0.378, 26.2)	(–0.098, 0.380, 28.6)	(–0.184, 0.384, 30.2)
2	(0.122, 0.381, 24.4)	(0.011, 0.378, 26.6)	(–0.094, 0.380, 28.6)
3	(0.134, 0.381, 24.1)	(0.069, 0.379, 25.4)	(–0.035, 0.378, 27.4)
4	(0.136, 0.381, 24.1)	(0.101, 0.380, 24.8)	(0.007, 0.378, 26.6)
5	(0.137, 0.381, 24.1)	(0.118, 0.381, 24.4)	(0.038, 0.378, 26.0)
	.....	.....	.....
10	(0.137, 0.381, 24.1)	(0.136, 0.381, 24.1)	(0.111, 0.380, 24.6)

The convergence is fast for dot features but is slow for contour and area features. However, the applicable range of positions, orientations and motions is still wider than for determining the three motion parameters simultaneously.

5.5 Error Sensitivity

The correspondenceless approach based on feature extraction is in general expected to be more robust than the use of correspondence; since the feature values are computed by summation or integration, small errors in the positions of feature points or lines tend to be canceled or diluted in the final values of the features.

In the following example, uniformly distributed random noise of 10% of the size l_0 and l_0' of the two regions S, S' (cf. eqns (5.1)) is added to the coordinate values of the corner points in the example of Fig. 9. The following result is one example. After the initial adjustment described by eqn (5.4), the estimate of the surface parameters is (–0.408, 0.408, 29.7). The scheme and interpretation are the same as in the previous example.

	corner features	contour features	area features
1	(0.095, 0.380, 26.0)	(–0.100, 0.380, 28.2)	(–0.144, 0.382, 29.7)
2	(0.167, 0.383, 23.9)	(–0.003, 0.378, 26.2)	(–0.044, 0.378, 27.7)
3	(0.178, 0.384, 23.6)	(0.042, 0.378, 25.3)	(0.021, 0.379, 26.4)
4	(0.179, 0.384, 23.5)	(0.065, 0.379, 24.8)	(0.065, 0.379, 25.5)
5	(0.179, 0.384, 23.5)	(0.075, 0.379, 24.6)	(0.097, 0.380, 24.9)
	.....	.....	.....
10	(0.180, 0.384, 23.5)	(0.084, 0.379, 24.4)	(0.166, 0.383, 23.5)

The results seem quite reasonable. Errors are about 30% in p and 1% in q and r. We must say that the motion is quite accurately computed, since the computed motion parameters (a,c,Ω) after the tenth iteration are (14.9, 15.0, –32.4) for dot features, (14.8, 14.7, –27.0) for contour features and (14.9, 14.8, –31.6) for area features. The rotation angle Ω is measured in degrees. Since the motion is horizontally constrained, vertical translation b is identically zero and the rotation axis is always (0,1,0). The true parameters are (15.0, 15.0, –30.0), so that errors are about 10% in Ω and about 2% in a and c. Thus, the error in the rotation angle is magnified to some extent when expressed in terms of p.

If uniformly distributed 10% noise is further added to the initial values of the surface parameters p, q, r, the result in a typical case is as follows. The motion parameters (p,q,r) are perturbed from the true values (–0.408, 0.408, 10.0) into (–0.395, 0.387, 9.9). After the initial adjustment described by eqn (5.4), the estimate of the surface parameters is (–0.395, 0.387, 29.2).

	corner features	contour features	area features
1	(0.088, 0.361, 25.8)	(–0.106, 0.362, 27.9)	(–0.145, 0.363, 29.3)
2	(0.157, 0.364, 23.8)	(–0.014, 0.360, 26.1)	(–0.047, 0.360, 27.5)
3	(0.168, 0.365, 23.5)	(0.029, 0.360, 25.2)	(0.016, 0.360, 26.2)
4	(0.169, 0.365, 23.4)	(0.049, 0.360, 24.8)	(0.059, 0.360, 25.3)
5	(0.170, 0.365, 23.4)	(0.058, 0.360, 24.6)	(0.090, 0.361, 24.7)
	.....	.....	.....
10	(0.170, 0.365, 23.4)	(0.067, 0.360, 24.5)	(0.155, 0.364, 23.4)

Errors are about 30% in p and about 5% in q and r. The motion parameters (a,c,Ω) after the tenth iteration are (14.8, 14.9, –31.1) for dot features, (14.7, 14.6, –25.4) for contour features and (14.8, 14.7, –30.4) for area features, so that the motion parameters themselves are quite accurately computed in spite of the existence of noise.

Fig. 10(a) is an example of a sequence of object images undergoing a horizontally constrained motion. Again, the plane constraining the motion is parallel to the zx-plane. For each image, 10% uniformly distributed random noise of its size (cf. eqn (5.1)) is added to the coordinates of the corner points. The true images are indicated by dotted line. Fig. 10(b) shows computed surface positions viewed from above (i.e., along the y-axis). A solid line is used for the true positions, a broken line for positions computed from the noisy images in Fig. 10(a), and a dotted line for positions computed by adding an additional 10% noise to the initial estimates of the surface parameters. If we use the noiseless images of Fig. 10(a), the true positions are obtained.

The computation is based on dot features, using the scheme of alternate translations and rotations. The iterations are stopped after ten stages. The initial image (the left-most one in Fig. 10(a)) is always compared with the image at each stage, so that error does not accumulate. In view of the fact that we are comparing a 10% noisy image with another 10% noisy image (see Fig. 10(a)), the recovery is fairy good and is almost unaffected by the noise in the initial estimates of the surface parameters.

6. DISCUSSION

6.1 Applicability of the Scheme

As seen in the examples in the previous section, our scheme can be applied to a very large motion. In general, convergence is fast for small motions. The iterations do not converge if the motion is extremely large (especially for rotations through large angles). However, the alternate application of the translational and rotational schemes can greatly extend the applicable range for the position and motion, covering almost all motions we may encounter in practical situations, although convergence may become somewhat slow. The assumption of horizontally constrained motion also extends the applicable range and makes the convergence faster. One general conclusion is that all the motion parameters can be determined simultaneously if the object is known to be near the camera (i.e., the distance to the object is not large compared with the focal length). Howerver, the scheme of alternate translations and rotations should be used if the object is known to be far away from the camera.

6.2 Shape and Choice of Features

Although we used polygonal regions in our examples, application of the present scheme is not limited to polygonal regions: it can be applied to arbitrary shapes, and the contours can be general curves. From our experiments, however, convergence seems to be better for simple convex shapes than for complicated non-convex shapes.

Among the three types of features (dot features, line features, area features), dot features seem to be the most favorable in convergence, although the difference is not very large. However, since line features (especially contours) are more stable than feature points (such as corner points) and line integrals are computationally easy to handle, the use of line features seems to be the best fit to practical applications.

6.3 Stability of Computation

As shown in the example in the previous section, the computation is very robust to noise, especially as regards the motion parameters. The example given by Tsai and Huang [29] shows that their result is unreliable even at a 1 to 2% noise level. The stability of our scheme may be ascribed to the following two reasons. First, we assume the initial position and compute only the motion parameters, while schemes like that of Tsai and Huang [29] attempt to determine the object position and motion simultaneously. In other words, the dimensionality of our solution space is smaller; the number of unknowns is six if the motion is unconstrained and is three if the motion is horizontally constrained. Second, we compute "features" of a planar region, which are macroscopic quantities obtained by summation or integration. Hence, independent noise at different points on the image plane tends to be canceled.

In general, there are many factors other than random noise that affect motion detection - illumination change, reflectance change, occlusion, etc. In view of these factors, the use of gray levels is not adequate, as discussed earlier. Even if images are represented by feature points and lines, corresponding feature points may be missing in some frames due to noise, illumination change, reflectance change, occlusion, etc. In contrast, region-to-region correspondence is much more stable. Our method is applicable unless the object does not have a planar face at all or all of the planar faces are occluded at the same time, which is very unlikely.

6.4 Face Identification

In our method, no knowledge of point-to-point correspondence between different frames is necessary. Instead, we must determine region-to-region correspondence. This is generally easier than establishing point-to-point correspondence. Suppose the object has planar faces, but it need not be a polyhedron. Take a car for example. Our method is applicable if we can identify planar windowpanes.

Assuming that the two given images are appropriately segmented, we select one planar region from one image. All we need to do is determine the corresponding region in the other image. We can introduce various "measures of similarity" to decide on the correspondence. Even if they are unavailable, we can resort to a brute force method. If there are N candidates for the corresponding region (usually N is not very large), there exist N possible correspondences. We can apply our method to all of these possibilities to test for the existence of a motion which correctly maps one region onto the other.

An important fact is that only one region-to-region correspondence is sufficient to determine the motion uniquely. If we use point-to-point correspondence, at least three correspondence pairs are necessary. The brute force method becomes inefficient in this case; if there are n feature points (usually n is much larger than N) and if we choose three points from one image arbitrarily, the number of possible correspondences is $n(n-1)(n-2)$. Granted that efficient methods for point-to-point correspondence detection are available, detection of region-to-region correspondence is much easier due to the more limited possibilities.

6.5 Application to Object Recognition

A major restriction on the present method is that we must measure the initial position of the object, say, by stereo or range sensing. However, this restriction also applies to the correspondenceless approaches of Lin et al. [22] and Aloimonos and Basu [3] using two cameras. While their methods require two cameras at each stage, our method requires position measurement (stereo or range sensing) only at the initial stage. This is a great advantage, since accurate measurement of position by stereo or range sensing usually takes time. Besides, their methods are based on corresponding feature points, while our method is not limited to the use of feature points; our method works for other types of features such as line segments and contours.

Once the initial position is known, we can compute the subsequent motion successively from a given image sequence. Errors may accumulate at each stage, but this is not a serious problem because our method applies to finite motion; any image frame can be compared, in principle, with the initial frame.

This fact can be used to advantage in model-based object recognition. Suppose we want to know the position and orientation of an object in the scene when the true size and shape of the object are known and are stored in a database. Since the true size and shape are known, we can easily generate the image of the object placed in a known position by, say, a computer graphics technique. Then, the position and orientation of the object are determined by computing the "motion" between this reference image and the observed image. The computation becomes efficient if several reference images and feature values are precomputed and stored in the same database.

Acknowledgment. The author thanks Prof. Azriel Rosenfeld and Prof. Larry S. Davis of the University of Maryland for helpful comments and discussions.

REFERENCES

1. G. Adiv, Determining 3-D motion and structure from optical flow generated by several moving objects, *IEEE Trans. Pattern Anal. Machine Intell.*, **PAMI 7**-4, 1985, 384 - 401.
2. G. Adiv, Inherent ambiguities in recovering 3-D motion and structure from a noisy flow field, *Proc. DARPA Image Understanding Workshop*, Miami Beach, FL, 1985, pp. 399 - 412.
3. J. Aloimonos and A. Basu, Shape and 3-D motion from contour without point to point correspondences: general principles, *Proc. IEEE Conf. Comput. Vision Pattern Recog.* Miami Beach, FL, 1986, pp. 518 -527.
4. S. Amari, Invariant structures of signal and feature spaces in pattern recognition problems, *RAAG Memoirs*, **4**, 1968, 553 - 566.
5. S. Amari, Feature spaces which admit and detect invariant signal transformations, *Proc. 4th Int. Joint Conf. Pattern Recog.*, Tokyo, 1978, pp. 452 - 456.
6. P. Anandan, *Computing Optical Flow from Two Frames of an Image Sequence*, COINS Technical Report 86-16, Department of Computer and Information Science, University of Massachusetts at Amherst, April, 1986.
7. P. Anandan and R. Weiss, *Introducing a Smoothness Constraint in a Matching Approach for the Computation of Displacement Fields*, COINS Technical Report 85-38, Department of Computer and Information Science, University of Massachusetts at Amherst, December, 1985.
8. A. Bandopadhay and R. Dutta, Measuring image motion in dynamic images, *Proc. IEEE Workshop on Motion: Representation and Analysis*, Charleston, SC, May, 1986, pp. 67 - 72.
9. B. F. Buxton, D. W. Murray, H. Buxton and N. S. Williams, Structure-from-motion algorithms for computer vision on an SIMD architecture, *Comp. Phys. Comm.*, **37**, 1985, 273 - 280.
10. D. Cyganski and J. A. Orr, Applications of tensor theory to object recognition and orientation determination, *IEEE Trans. Pattern Anal. Machine Intell.*, **PAMI 7**-6, 1985, 662 - 673.
11. W. Enkelmann, Investigation of multigrid algorithms for the estimation of optical flow fields in image sequences, *Proc. IEEE Workshop on Motion: Representation and Analysis*, Charleston, SC, May, 1986, pp. 81 -87.
12. B. K. P. Horn and B. G. Schunck, Determining optical flow, *Artif. Intell.*, **17**, 1981, 185 - 203.
13. K. Kanatani, Detection of surface orientation and motion from texture by a stereological technique, *Artif. Intell.*, **23**, 1984, 213 - 237.
14. K. Kanatani, Tracing planar surface motion from projection without knowing the correspondence, *Comput. Vision Graphics Image Process.*, **29**, 1984, 1 - 12.
15. K. Kanatani, Detecting the motion of a planar surface by line and surface integrals, *Comput. Vision Graphics Image Process.*, **29**, 1984, 13 - 22.
16. K. Kanatani, Structure and motion without correspondence: general principle, *Proc. 9th Int. Joint. Conf. Artif. Intell.*, Los Angeles, CA, 1985, pp. 886 - 888.
17. K. Kanatani, Structure and motion without correspondence: general principle, *Proc. DARPA Image Understanding Workshop*, Miami Beach, FL, 1985, pp. 107 - 116.
18. K. Kanatani, Structure and motion from optical flow under orthographic projection, *Comput. Vision Graphics Image Process.* (to appear).
19. K. Kanatani, Structure and motion from optical flow under perspective projection, *Comput. Vision Graphics Image Process.* (to appear).
20. K. Kanatani and T.-C. Chou, Shape from texture: general principle, *Proc. IEEE Conf. Comput. Vision Pattern Recog.* Miami Beach, FL, 1986, pp. 578 - 583.
21. R. Kories and G. Zimmermann, A versatile method for the estimation of displacement vector fields from image sequences, *Proc. IEEE Workshop on Motion: Representation and Analysis*, Charleston, SC, May, 1986, pp. 101 - 106.
22. Z.-C. Lin, T. S. Huang, S. D. Blostein, H. Lee and E. A. Margerum, Motion estimation from 3-D point sets with and without correspondences, *Proc. IEEE Conf. Comput. Vision Pattern Recog.* Miami Beach, FL, 1986, pp. 194 - 201.
23. H. C. Longuet-Higgins, A computer algorithm for reconstructing a scene from two projections, *Nature*, **239**, 1981, 133 - 135.
24. H. C. Longuet-Higgins, The visual ambiguity of a moving plane, *Proc. R. Soc. Lond.*, **B-223**, 1984, 165 - 175.
25. H.-H. Nagel, Representation of moving rigid objects based on visual observations, *Computer*, **14**-8, 1981, 29 - 39.
26. J. M. Prager and M. A. Arbib, Computing the optic flow: the MATCH algorithm and prediction, *Comput. Vision Graphics Image Process.*, **24**, 1983, 271 - 304.
27. A. Rosenfeld and A. C. Kak, *Digital Picture Processing*, Vol. 2, Academic Press, New York, NY, 1982.
28. M. Subbarao and A. M. Waxman, Closed form solution to image flow equations for planar surface in motion, *Comput. Vision Graphics Image Process.* (to appear).
29. R. Y. Tsai and T. S. Huang, Uniqueness and estimation of three-dimensional motion parameters of rigid objects with curved surfaces, *IEEE Trans. Pattern Anal. Machine Intell.*, **PAMI 6**-1, 1984, 13 - 27.
30. S. Ullman, *The Interpretation of Visual Motion*, MIT Press, Cambridge, MA. 1979.
31. A. M. Waxman and S. Ullman, Surface structure and three-dimensional motion from image flow kinematics, *Int. J. Robotics Res.*, **4**-3, 1985, 72 - 94.
32. A. M. Waxman and K. Wohn, Contour evolution, neighborhood deformation, and global image flow: planar surfaces in motion, *Int. J. Robotics Res.*, **4**-3, 1985, 95 - 108.

APPENDIX A. CONJUGATE FEATURE TRANSFORMATION

A.1 Conjugate Transformation of Features

The principle of our method is to iteratively transform one region into another. If a region is characterized by a small number of parameters, say the coordinates of the corners of a polygonal region, the transformation is easy; only these parameter values need be altered. However, if the region contains a large amount of information, e.g., when the surface is finely textured, transformation of the image requires large amounts of computation time and memory space. If this is the case, we need not transform the image; all we need is the *transformed feature values*, not the transformed image itself.

Define the *image transformation operator* T_A by

$$\tilde{X}(x,y)=T_A X(x,y), \tag{A.1}$$

where A is the matrix prescribing the transformation of eqns (2.5). Since the inverse transformation of eqns (2.5) is given by replacing matrix $A=(A_{ij})$ by its inverse $A^{-1}=(A_{ij}^{-1})$, operator T_A is defined as follows:

Definition 1 (Image Transformation Operator).

$$T_A X(x,y)=X(f\frac{A_{11}^{-1}x+A_{12}^{-1}y+A_{13}^{-1}f}{A_{31}^{-1}x+A_{32}^{-1}y+A_{33}^{-1}f}, f\frac{A_{21}^{-1}x+A_{22}^{-1}y+A_{23}^{-1}f}{A_{31}^{-1}x+A_{32}^{-1}y+A_{33}^{-1}f}). \tag{A.2}$$

This equation states that the value of the transformed image $T_A X$ at (x,y) is given by the value of the original image X at the point obtained by applying the inverse transformation to the point (x,y).

Now, define the *conjugate transformation operator* T_A^* by

Definition 2 (Conjugate Transformation Operator).

$$T_A^* F[X]=F[T_A X]. \tag{A.4}$$

This equation states that the value of feature $F[\,.\,]$ of the transformed image X is obtained as the value of the transformed feature $T_A^* F[\,.\,]$ of the original image X. An essential fact is that operator T_A^* acts on a given *functional* $F[\,.\,]$ by $T_A^* F[\,.\,]=F[T_A(\,.\,)]$ *without any regard to particular images*. Hence, we need not transform the image according to eqn (A.2); measuring feature $F[X]$ of the transformed image $\tilde{X}(x,y)$ is equivalent to measuring feature $T_A^*[X]$ of the original image $X(x,y)$. Hence, once we know how operator T_A^* acts on functionals, the computation can always be performed on the original image.

Consequently, the motion parameters $\tilde{a}$, $\tilde{b}$, $\tilde{c}$, $\tilde{\Omega}_1$, $\tilde{\Omega}_2$, $\tilde{\Omega}_3$ of the next step are given by soiving

$$\begin{bmatrix} T_A^* C_i^{(1)}[X] & T_A^* C_i^{(1)}[X] & T_A^* C_i^{(1)}[X] & T_A^* C_{\Omega_1}^{(1)}[X] & T_A^* C_{\Omega_2}^{(1)}[X] & T_A^* C_{\Omega_3}^{(1)}[X] \\ T_A^* C_i^{(2)}[X] & T_A^* C_i^{(2)}[X] & T_A^* C_i^{(2)}[X] & T_A^* C_{\Omega_1}^{(2)}[X] & T_A^* C_{\Omega_2}^{(2)}[X] & T_A^* C_{\Omega_3}^{(2)}[X] \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \end{bmatrix} \begin{bmatrix} \tilde{a} \\ \tilde{b} \\ \tilde{c} \\ \tilde{\Omega}_1 \\ \tilde{\Omega}_2 \\ \tilde{\Omega}_3 \end{bmatrix}$$

$$= \begin{bmatrix} F^{(1)}[X']-T_A^* F^{(1)}[X] \\ F^{(2)}[X']-T_A^* F^{(2)}[X] \\ \cdots \\ \cdots \end{bmatrix} \tag{A.5}$$

A.2 Change of Variables

From the discussion in the above, it remains to determine how the conjugate transformation operator T_A^* acts on particular feature functionals. For that purpose, we give here some mathematical preliminaries in preparation for the subsequent discussion.

Consider two points (x,y) and $(x+dx,y+dy)$ infinitesimally far apart from each other on image X, and let $(\tilde{x},\tilde{y})$, $(\tilde{x}+d\tilde{x},\tilde{y}+d\tilde{y})$ be the corresponding points on the transformed image $\tilde{X}$. By differentiating the equations of transformation

$$\tilde{x}=f\frac{A_{11}x+A_{12}y+A_{13}f}{A_{31}x+A_{32}y+A_{33}f}, \qquad \tilde{y}=f\frac{A_{21}x+A_{22}y+A_{23}f}{A_{31}x+A_{32}y+A_{33}f}, \tag{A.6}$$

we see that differentials $d\tilde{x}$, $d\tilde{y}$ are related to differentials dx, dy by

$$d\tilde{x}=\frac{\partial\tilde{x}}{\partial x}dx+\frac{\partial\tilde{x}}{\partial y}dy, \qquad d\tilde{y}=\frac{\partial\tilde{y}}{\partial x}dx+\frac{\partial\tilde{y}}{\partial y}dy, \tag{A.7}$$

$$\begin{bmatrix} \partial\tilde{x}/\partial x & \partial\tilde{x}/\partial y \\ \partial\tilde{y}/\partial x & \partial\tilde{y}/\partial y \end{bmatrix}=-\frac{1}{A_{31}x+A_{32}y+A_{33}f}\begin{bmatrix} A_{31}\tilde{x}-A_{11}f & A_{32}\tilde{x}-A_{12}f \\ A_{31}\tilde{y}-A_{21}f & A_{32}\tilde{y}-A_{22}f \end{bmatrix} \tag{A.8}$$

Let us consider a smooth curve L on the xy-plane. If $(x(s),y(s))$ is its parametric form, where s is the arc length along the curve L, the unit tangent vector $(n_1(s),n_2(s))$ to the curve L is given by

$$n_1(s)=\frac{\partial x}{\partial s}, \qquad n_2(s)=\frac{\partial y}{\partial s}. \tag{A.9}$$

Let $\tilde{L}$ be the curve on the $\tilde{x}\tilde{y}$-plane corresponding to the curve L on the xy-plane, and let $(\tilde{x}(\tilde{s}),\tilde{y}(\tilde{s}))$ be its parametric form, where $\tilde{s}$ is the arc length along the curve $\tilde{L}$. Substituting eqns (A.7) and (A.8) in $d\tilde{s}^2=d\tilde{x}^2+d\tilde{y}^2$, we obtain

$$d\tilde{s}^2=Edx^2+2Fdxdy+Gdy^2, \tag{A.10}$$

where

$$E\equiv\frac{A_{31}{}^2(\tilde{x}^2+\tilde{y}^2)-2A_{31}f(A_{11}\tilde{x}+A_{21}\tilde{y})+(A_{11}{}^2+A_{21}{}^2)f^2}{(A_{31}x+A_{32}y+A_{33}f)^2},$$

$$F\equiv\frac{A_{31}A_{32}(\tilde{x}^2+\tilde{y}^2)-(A_{11}A_{32}+A_{21}A_{31})f\tilde{x}-(A_{21}A_{32}+A_{22}A_{31})f\tilde{y}+(A_{11}A_{12}+A_{21}A_{22})f^2}{(A_{31}x+A_{32}y+A_{33}f)^2} \tag{A.11}$$

$$G\equiv\frac{A_{32}{}^2(\tilde{x}^2+\tilde{y}^2)-2A_{32}f(A_{12}\tilde{x}+A_{22}\tilde{y})+(A_{12}{}^2+A_{22}{}^2)f^2}{(A_{31}x+A_{32}y+A_{33}f)^2}.$$

Combining this with eqns (A.9), we have

$$d\tilde{s}=\Gamma(s)ds, \tag{A.12}$$

where

$$\Gamma(s)\equiv\sqrt{E(x(s),y(s))n_1(s)^2+2F(x(s),y(s))n_1(s)n_2(s)+G(x(s),y(s))n_2(s)^2} \tag{A.13}$$

The unit tangent vector to the curve $\tilde{L}$ is given by $(\tilde{n}_1,\tilde{n}_2)$. Since both (n_1,n_2) and $(\tilde{n}_1,\tilde{n}_2)$ are unit vectors, they are related by

$$\begin{bmatrix} \tilde{n}_1 \\ \tilde{n}_2 \end{bmatrix}=\frac{1}{\sqrt{En_1{}^2+2Fn_1n_2+Gn_2{}^2}}\begin{bmatrix} \partial\tilde{x}/\partial x & \partial\tilde{x}/\partial y \\ \partial\tilde{y}/\partial x & \partial\tilde{y}/\partial y \end{bmatrix}\begin{bmatrix} n_1 \\ n_2 \end{bmatrix}. \tag{A.14}$$

Now, we define the conjugate transformation T_A^* of the unit tangent vector (n_1,n_2) by

$$T_A^* n_1\equiv\frac{1}{\sqrt{En_1{}^2+2Fn_1n_2+Gn_2{}^2}}(\frac{\partial\tilde{x}}{\partial x}n_1+\frac{\partial\tilde{x}}{\partial y}n_2), \tag{A.15}$$

$$T_A^* n_2 \equiv \frac{1}{\sqrt{En_1^2+2Fn_1n_2+Gn_2^2}}\left(\frac{\partial \tilde{y}}{\partial x}n_1+\frac{\partial \tilde{y}}{\partial y}n_2\right),$$

and the conjugate transformation T_A^* of a function $m(x,y)$ and its derivatives by

$$T_A^* m(x,y) \equiv m(\tilde{x}(x,y),\tilde{y}(x,y)), \tag{A.16}$$

$$T_A^* \frac{\partial m}{\partial x} \equiv \frac{\partial m}{\partial x}(\tilde{x}(x,y),\tilde{y}(x,y)), \quad T_A^* \frac{\partial m}{\partial y} \equiv \frac{\partial m}{\partial y}(\tilde{x}(x,y),\tilde{y}(x,y)). \tag{A.17}$$

Finally, consider the region $\tilde{S}$ on the $\tilde{x}\tilde{y}$-plane corresponding to a region S on the xy-plane. The area of $\tilde{S}$ is given by

$$\int_{\tilde{S}} d\tilde{x}d\tilde{y} = \int_S J(x,y)dxdy, \tag{A.18}$$

where $J(x,y)$ is the *Jacobian* defined by

$$J(x,y) = \begin{vmatrix} \partial\tilde{x}/\partial x & \partial\tilde{x}/\partial y \\ \partial\tilde{y}/\partial x & \partial\tilde{y}/\partial y \end{vmatrix} = \sqrt{E(x,y)G(x,y)-F(x,y)^2}$$

$$= f\frac{(A_{21}A_{32}-A_{22}A_{31})\tilde{x}+(A_{12}A_{31}-A_{11}A_{32})\tilde{y}+(A_{11}A_{22}-A_{21}A_{12})f}{(A_{31}x+A_{32}y+A_{33}f)^2}. \tag{A.19}$$

A.3 Conjugate Transformation Operator

The feature functionals for dot features have the form

$$I[X] = \sum_{P_i \in S} J(x_i,y_i,m(x_i,y_i),\frac{\partial m}{\partial x}(x_i,y_i),\frac{\partial m}{\partial y}(x_i,y_i)). \tag{A.20}$$

By Definition 2, the action of the conjugate transformation operator T_A^* is given by

$$T_A^* I[X] = I[T_A X]$$

$$= \sum_{\tilde{P}_i \in \tilde{S}} J(\tilde{x}_i,\tilde{y}_i,m(\tilde{x}_i,\tilde{y}_i),\frac{\partial m}{\partial x}(\tilde{x}_i,\tilde{y}_i),\frac{\partial m}{\partial y}(\tilde{x}_i,\tilde{y}_i)),$$

$$= \sum_{P_i \in S} J(\tilde{x}(x_i,y_i),\tilde{y}(x_i,y_i),T_A^* m(x_i,y_i),T_A^*\frac{\partial m}{\partial x}(x_i,y_i),T_A^*\frac{\partial m}{\partial y}(x_i,y_i)), \tag{A.21}$$

where $\tilde{P}_i(\tilde{x}_i,\tilde{y}_i)$ are the transformed positions of the dots.

The feature functionals for line features have the form

$$I[X] = \sum_{L_i \subset S} \int_{L_i} J(x,y,n_1,n_2,m(x,y),\frac{\partial m}{\partial x},\frac{\partial m}{\partial y})ds. \tag{A.22}$$

By Definition 2 and the rules of change of variables, the action of the conjugate transformation operator T_A^* is given by

$$T_A^* I[X] = I[T_A X]$$

$$= \sum_{\tilde{L}_i \subset \tilde{S}} \int_{\tilde{L}_i} J(\tilde{x},\tilde{y},\tilde{n}_1,\tilde{n}_2,m(\tilde{x},\tilde{y}),\frac{\partial m}{\partial x}(\tilde{x},\tilde{y}),\frac{\partial m}{\partial y}(\tilde{x},\tilde{y}))d\tilde{s},$$

$$= \sum_{L_i \subset S} \int_{L_i} J(\tilde{x}(x,y),\tilde{y}(x,y),T_A^* n_1,T_A^* n_2,T_A^* m(x,y),T_A^*\frac{\partial m}{\partial x},T_A^*\frac{\partial m}{\partial y})\Gamma(s)ds, \tag{A.23}$$

The feature functionals for area features have the form

$$I[X] = \int\int_S J(x,y,m(x,y),\frac{\partial m}{\partial x},\frac{\partial m}{\partial y})dxdy. \tag{A.24}$$

The action of the conjugate transformation operator T_A^* is given by

$$T_A^* I[X] = I[T_A X]$$

$$= \int\int_{\tilde{S}} J(\tilde{x},\tilde{y},m(\tilde{x},\tilde{y}),\frac{\partial m}{\partial x}(\tilde{x},\tilde{y}),\frac{\partial m}{\partial y}(\tilde{x},\tilde{y}))d\tilde{x}d\tilde{y}$$

$$= \int\int_S J(\tilde{x}(x,y),\tilde{y}(x,y),T_A^* m(x,y),T_A^*\frac{\partial m}{\partial x},T_A^*\frac{\partial m}{\partial y})J(x,y)dxdy, \tag{A.25}$$

where $\tilde{S}$ is the transformed region and $J(x,y)$ is the Jacobian given by the determinant of eqn (A.8).

APPENDIX B. GRAY-LEVEL IMAGES

As discussed earlier, gray-level images are not adequate in general for stable feature detection. However, if the value $X(x,y)$ of "image intensity" can be assumed to be inherent to the object and is not influenced by the viewing orientation, for example if it is chromaticity, we can also use the direct filtering proposed by Amari [3, 4] in the form

$$F[X] = \int\int_S m(x,y)X(x,y)dxdy. \tag{B.1}$$

If each point (x,y) is displaced by $(\Delta x,\Delta y)$, the difference $\Delta F[X]$ of the feature values is given by

$$\Delta F[X] = \int\int_S [\frac{\partial m}{\partial x}\Delta x+\frac{\partial m}{\partial y}\Delta y+(\frac{\partial \Delta x}{\partial x}+\frac{\partial \Delta y}{\partial y})m]X(x,y)dxdy+\cdots, \tag{B.2}$$

where ... again denotes higher order terms in Δx, Δy and their derivatives. Substituting the equation of optical flow (2.10), we find

$$C_{u_0}[X] = \int\int_S \frac{\partial m}{\partial x}Xdxdy, \quad C_{v_0}[X] = \int\int_S \frac{\partial m}{\partial y}Xdxdy,$$

$$C_A[X] = \int\int_S [m+x\frac{\partial m}{\partial x}]Xdxdy, \quad C_B[X] = \int\int_S y\frac{\partial m}{\partial x}Xdxdy,$$

$$C_C[X] = \int\int_S x\frac{\partial m}{\partial y}Xdxdy, \quad C_D[X] = \int\int_S [m+y\frac{\partial m}{\partial y}]Xdxdy, \tag{B.3}$$

$$C_E[X] = \int\int_S [3xm+x^2\frac{\partial m}{\partial x}+xy\frac{\partial m}{\partial y}]Xdxdy,$$

$$C_F[X] = \int\int_S [3ym+xy\frac{\partial m}{\partial x}+y^2\frac{\partial m}{\partial y}]Xdxdy.$$

If we set $X(x,y)=1$, all the above reduces to eqns (4.11).

The functionals here defined as area integrals have the form

$$I[X] = \int\int_S J(x,y,m(x,y),\frac{\partial m}{\partial x},\frac{\partial m}{\partial y})X(x,y)dxdy. \tag{B.4}$$

The action of the conjugate transformation operator T_A^* is given by

$$T_A^* I[X] = I[T_A X]$$

$$= \int\int_{\tilde{S}} J(\tilde{x},\tilde{y},m(\tilde{x},\tilde{y}),\frac{\partial m}{\partial x}(\tilde{x},\tilde{y}),\frac{\partial m}{\partial y}(\tilde{x},\tilde{y}))\tilde{X}(\tilde{x},\tilde{y})d\tilde{x}d\tilde{y}$$

$$= \int\int_S J(\tilde{x}(x,y),\tilde{y}(x,y),T_A^* m(x,y),T_A^*\frac{\partial m}{\partial x},T_A^*\frac{\partial m}{\partial y})X(x,y)J(x,y)dxdy. \tag{B.5}$$

where $\tilde{S}$ is the transformed region and $J(x,y)$ is the Jacobian given by the determinant of eqn (A.8).

Thus, if we take six independent functions $m_i(x,y)$, $i=1,\ldots,6$, we obtain the corresponding features $F^{(i)}[X]$ in the form of eqn (B.1), and $C_u^{(i)}[X]$, $C_v^{(i)}[X]$, $C_c^{(i)}[X]$, $C_{\Omega_1}^{(i)}[X]$, $C_{\Omega_2}^{(i)}[X]$, $C_{\Omega_3}^{(i)}[X]$, $i=1,\ldots,6$, are given by eqns (3.4). Hence, the motion is recovered by the procedure described in Section 3.

FIGURES

Fig. 1 A point (X,Y,Z) on a plane having equation $z=px+qy+r$ is projected to point (x,y) on the image plane by perspective projection from the viewpoint $(0,0,-f)$. The motion of the plane is specified relative to a reference point (X_0,Y_0,Z_0) taken on it.

Fig. 2 A rigid motion is specified by translation (a,b,c) of a reference point and rotation R around it.

Fig. 3 Composition of two rigid motions. Translations are added, and rotations are multiplied.

Fig. 4 Scheme of initial adjustment by size comparizon.

Fig. 5 An example of unconstrained motion.

Fig. 6 An example of unconstrained motion.

Fig. 7 Horizontally constrained motion.

Fig. 8 An example of horizontally constrained motion.

Fig. 9 An example of horizontally constrained motion.

Fig. 10 (a) An image sequence of a moving planar surface undergoing a horizontally constrained motion. For each image, random noise of 10% of its size is added to the corner positions. The true images are indicated by a dotted line. (b) Computed surface positions viewed from above (i.e., along the y-axis). A solid line is used for true positions, a broken line for positions computed from the noisy images in Fig. 10(a), and a dotted line for positions computed by adding additional 10% noise to the initial estimates of the motion parameters.

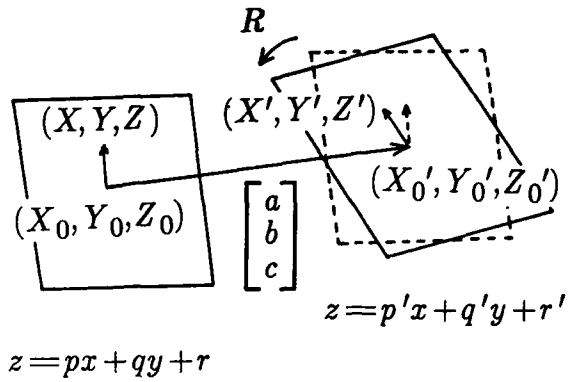


Fig. 2

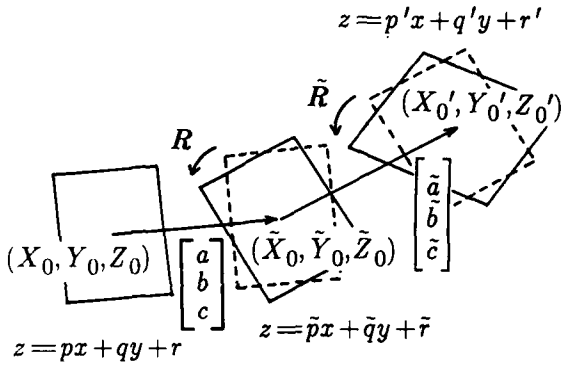


Fig. 3

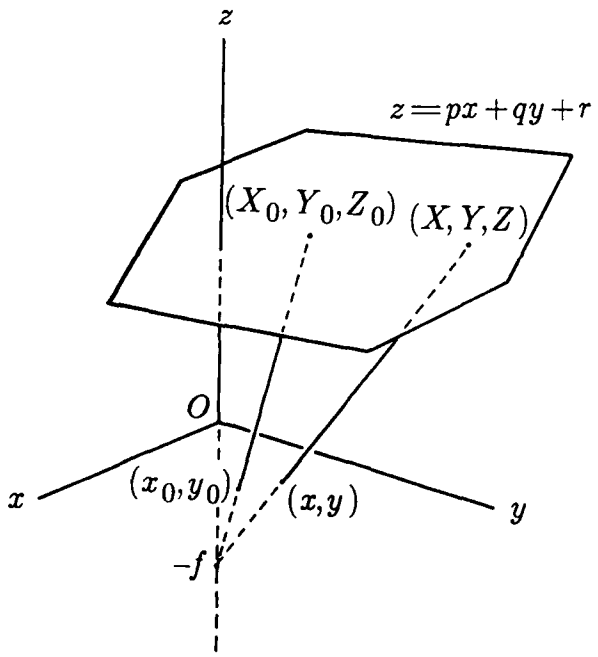


Fig. 1

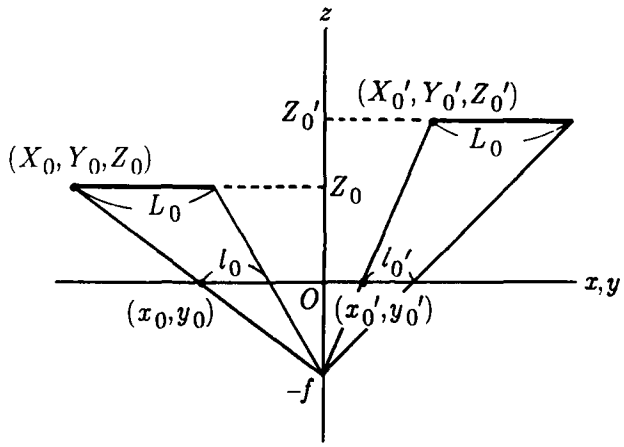


Fig. 4

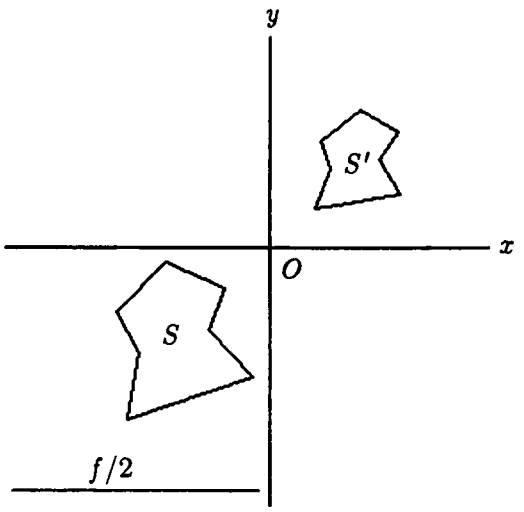


Fig. 5

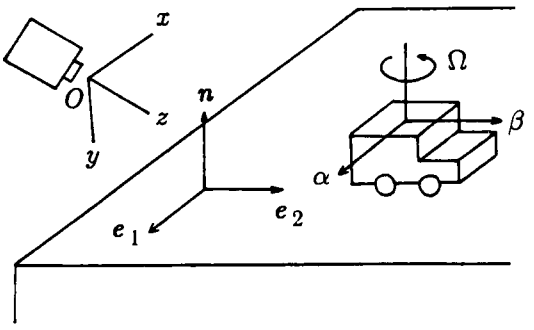


Fig. 7

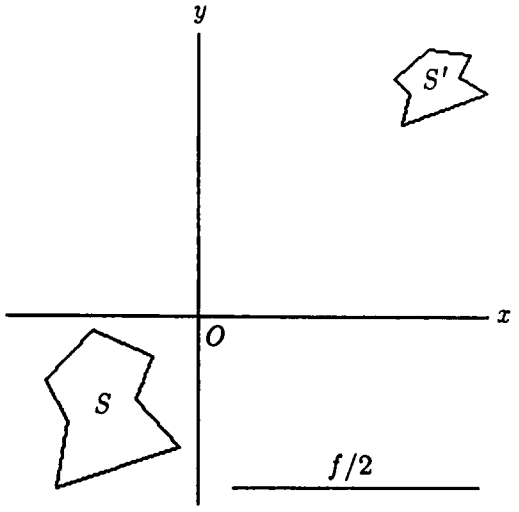


Fig. 6

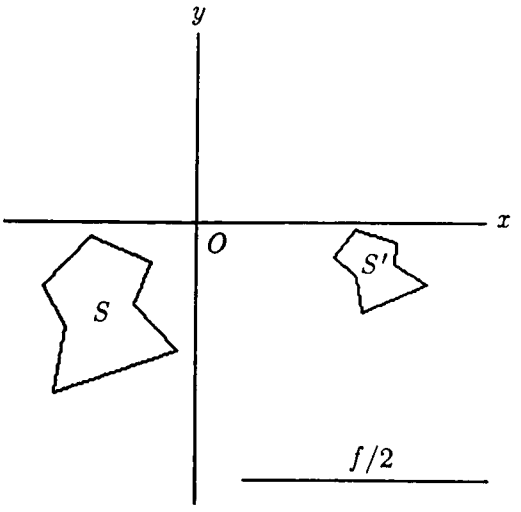


Fig. 8

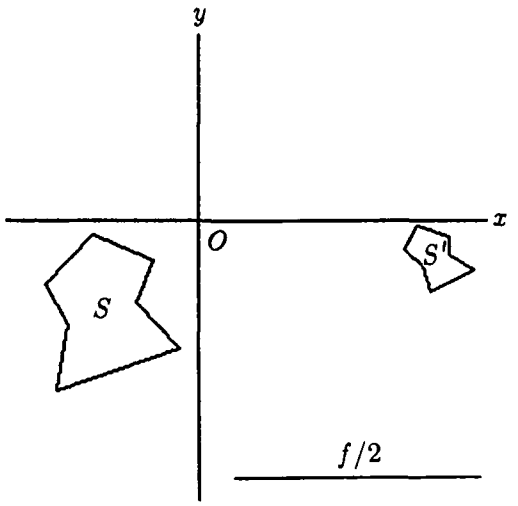


Fig. 9

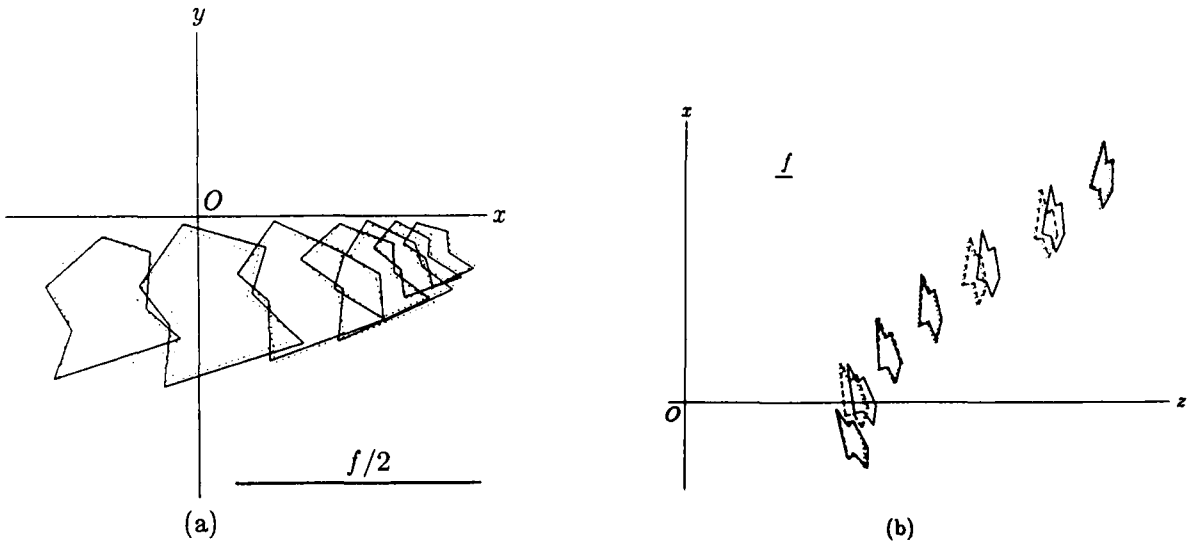


Fig. 10

A Unified Perspective on Computational Techniques for the Measurement of Visual Motion*

P. Anandan†
Computer and Information Science Department
University of Massachusetts
Amherst, Ma 01003

ABSTRACT

A unified hierarchical computational framework is developed for the determination of dense displacement fields from a pair of images. This framework incorporates the separation of computations according to scale, a coarse-to-fine control strategy, the explicit use of direction-dependent confidence measures, and a rigorous formulation of a smoothness constraint on image motion. The recent hierarchical extensions by Glazer and Enkelmann of the well known gradient-based techniques of Horn and Schunck and Nagel, and our own matching technique, which is based on the minimization of the sum-of-squared-differences (SSD) of the intensities, are all shown to be completely consistent with our framework. It is also shown that the gradient-based techniques can be regarded as mathematical limits of the matching technique. Since our framework allows the explicit identification of the various components that are necessary for the successful measurement of motion, the development of a single robust system which incorporates the best implementation choices for each component is within reach.

A modified version of our framework is proposed, in which the computations are separated according to scale and orientation. This new approach unifies the recently developed spatio-temporal energy models with the gradient-based and the matching techniques, and appears to be biologically feasible and ideally suited for connectionist models of computation.

1 Introduction

1.1 Background

This paper describes an attempt to develop a unified computational framework for the measurement of visual motion from a sequence of images. In computer vision, the most popular approach for motion analysis has been to measure image-motion prior to any determination of geometric structures. These measurements are then used to recover the 3-dimensional structure of the environment and to determine the relative 3-D motion between the camera and the objects in the scene. In this sense, the measurement of visual motion is treated synonymously with the measurement of image-motion.

The research effort in computer vision for the measurement of image-motion has focused primarily on (a) the determination of instantaneous image-velocities by using a "gradient-based" approach, or (b) the determination of displacements of points between successive frames using a matching approach. On the other hand, recent developments in psychophysics have focused on the determination of the speed and the direction of motion of a one-dimensional visual signal.

The research on the interpretation of motion [2,3,4] indicates that a dense and reliable displacement field may be necessary for the successful determination of the structure of the environment. The most effective techniques for computing dense displacement-fields seem to be those which embed either a gradient-based or a matching approach in a hierarchical, multi-resolution scheme. Such techniques include the multi-resolution gradient-based approaches of Enkelmann [12] and Glazer [15], and our own multi-resolution matching technique, which will be denoted here as Anandan's technique[6,7].

Although the gradient-based and the matching approaches appear to be unrelated to each other, they can be unified under a single computational framework. Such a framework is described in this paper and shown to be consistent with the different current approaches for the measurement of motion. The framework is developed from theoretical considerations, and each of its major components is shown to be essential for the computation of dense displacement fields. Besides the unification of a wide range of current techniques, the framework also allows the reduction of the task of designing a working system to that of making proper implementation choices for the various components of the framework.

1.2 Framework overview

The key idea underlying our framework is the separation of computations according to scale. This idea is based on the following observation: usually, the large scale (or

*This research was supported by DARPA under grant N00014-82-K-0464

†Current Address: Department of Computer Science, AI project, Yale University, New Haven, CT 06520

low spatial-frequency) intensity variations can provide imprecise measurements over a large range of magnitudes of motion, while the small-scale (or high spatial-frequency) variations can provide more accurate measurements over a smaller range. This leads to three components of our framework: **spatial-frequency decomposition**, which is the method of separating the intensity variations according to scale, a local, parallel **match-criterion** within each scale, and a **control-strategy**, which is a method for controlling the measurement processes at the different scales and combining their results.

Although the scale-based separation of computation provides a useful principle for processing scenes containing large displacements, there will always be situations when an image area lacks sufficient local information for displacement computation at a particular scale. Also, since the image-displacement is a vector quantity, its reliability can vary according to direction. Therefore another essential component of our framework is a direction-dependent **confidence measure**. The presence of unreliable displacements also means that in order to obtain a dense displacement field, it may be necessary to propagate the reliable displacements to their less reliable neighbors. This leads to the last essential component of our framework: a **smoothness constraint**, which specifies the criterion for the propagation of reliable displacements.

A visual illustration of this framework is provided in Figure 1. The five major components mentioned in the above description are discussed in detail in section 2.

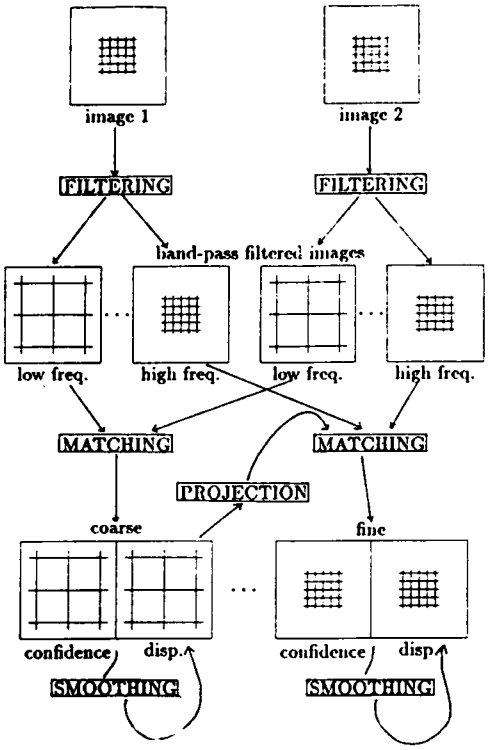


Figure 1: The hierarchical computational framework

1.3 Paper overview

The detailed description of our framework for computing dense displacement fields from a pair of images is contained in section 2. First, the goals of the displacement computation process are stated. This statement consists of specifying the nature of the input, the requirements on output, and the computational constraints for this process. This is followed by the detailed descriptions of the five components of the framework.

Section 3 contains a review of the principles underlying current techniques in computer vision and an outline of the three specific algorithms of Enkelmann, Glazer, and Anandan. Section 4 explains the relationship of these three techniques to our framework by identifying the implementation choices for the five components. In section 4, we also show that the two gradient-based techniques can be regarded as the mathematical limits of the matching approach. Section 5 contains a discussion of the problems involved in processing discontinuities in image motion, while Section 6 contains the results of applying Anandan's matching technique to a pair of real images, as a demonstration of the feasibility of the ideas contained in our framework.

As such, the spatio-temporal energy models [1] are not consistent with the computational framework described here. This is because the energy models have been formulated primarily for one-dimensional signals varying continuously over time. The use of the energy models for two-dimensional sequences would involve decomposing the input intensity image according to orientation. Such an approach can be found in the recent work of Heeger [18]. In section 7, we propose a modified form of our framework, in which all three categories of techniques are unified. This new approach appears to be biologically feasible and naturally suited for connectionist models of computation. In our own future research, we intend to explore possible methods of implementing this approach.

2 The Computational Framework

2.1 The computational goals

The goals of the process of computing image displacements are determined by three major factors: the nature of its input, the requirements on the output, and computational efficiency constraints. The input is a pair of digitized frames belonging to a discrete image sequence. The image displacements may be due to a general 3-dimensional motion of the camera or the independent motion of objects in the scene. The output should be a dense field of displacement vectors with associated confidence measures. All the computations must be pixel-parallel and use local

image information. Our motivations for choosing this set of goals are explained below.

The input – In typical video sequences, the inter-frame displacements are usually considerably larger than a pixel. Usually, due to the presence of independently moving objects, a single set of 3-D motion parameters will not be consistent with the entire image. The objects in the environment are assumed to be composed of continuous and opaque surfaces. It is also assumed that (i) within the image-area covered by a single surface, the displacement field varies smoothly, and (ii) the image-motion can be described as "locally translational", i.e., within a small area of the image, the displacement field can be approximated by a translational flow field. Discontinuities in image motion will occur at the boundaries of surfaces and objects.

The assumptions stated above are satisfied in a large class of real images. The major exceptions arise when the images contain transparent and/or "fence-like" surfaces, when the points on an object undergo non-rigid or chaotic motion, and when the amount of rotation is significant.

The output – The requirement that the output should be a dense displacement field with a confidence measure is derived from the conclusions of the various studies concerning the problem of extracting structure from motion [2,3,13,31,37]. These studies indicate that a large number of image displacements are necessary for the accurate determination of the structure of the environment. In addition, there should be an indication of the reliability of the displacements, so that inaccurate ones can be ignored. Adiv's approach [2] which appears to be at least partially successful at segmenting independently moving objects, assumes no *a priori* knowledge of the image-locations of such objects. Therefore, the density of the displacement vector field should be large uniformly across the image. These conclusions are further supported by his analysis of the inherent ambiguities in the interpretation of motion [3].

Computational considerations – The considerations of computational efficiency and ease of implementation suggest that the following three properties are desirable for all of our computations: *parallelism*, *uniformity*, and *localness*.

Parallelism simply means that it should be possible to perform all computations simultaneously at all locations on the image-plane. Uniformity implies that the process should be similar at all locations. It should be possible to describe any differences between the computations at different locations in terms of a few simple parameters. Localness means that the computations at any point on the image should be based on information local to that point.

2.2 Spatial-frequency decomposition

As noted briefly at the beginning of this paper, the key idea underlying our computational strategy is the separation of computation on the basis of scale. Intuitively, it is clear that while small scale intensity structures can be used to measure displacements over a short range, they may have many duplicate matches over a large range. This leads to ambiguities in the computation of the displacements. Therefore, in order to process large displacements, large scale intensity information must be used. However, a single displacement computed on the basis of a large scale intensity structure will be an average of the displacements over the area covered by that structure; hence, its accuracy will be low. Such a "smoothed" displacement field will also vary slowly over the image plane; hence, it can be sampled at a lower rate without loss of information.

These observations lead to the following principle: large-scale image structures can be used to measure displacements over a large range with low accuracy and at a low sampling density, while small-scale image structures can be used to measure displacements over a short range with higher accuracy and at a higher sampling density. An obvious way to enforce this principle is to decompose the image into its spatial-frequency components. Such a decomposition and the subsequent processing can be achieved by using a set of *spatial-frequency channels*.

Since the lower-frequency information can be sampled at a lower rate without any significant loss of information, the spatial-frequency decomposition process is usually accompanied by a corresponding reduction of resolution [9,38]. Such an approach leads to a pyramid representation of the spatial-frequency channels and fits naturally into a pyramid [21,32] or a processing-cone [17] architecture. However, since the final choice of a representation scheme depends on the type of hardware which is used, a pyramid representation is not an essential part of our framework.

2.3 The match-criterion

As noted earlier, the *match-criterion* is a method for determining the displacements within each channel. Since the displacement measured is small with respect to the scale of the intensity variations within a channel, a gradient-based approach can be used (see [12,15]). Alternatively, a correlation-matching approach [6,10,14] or a symbolic matching approach based on primitive tokens [16,23] can also be used. The separation of matching according to scale implies that the match-criterion should have a *scaling* property, i.e., the measurement processes within different channels should be scaled versions of each other. Note that such a scaling property is directly provided in a pyramid representation.

2.4 The control-strategy

The control-strategy determines how the measurement processes at different scales are controlled and how their results are combined. In our framework, the control strategy is based on a *spectral continuity* principle [16,24], which can be described as follows: For images of opaque surfaces, it can be assumed that the displacement estimates at corresponding image locations in the different channels must be similar because they are due to relative motion between the camera and the same environmental area. This means that at any image location, a displacement computed from a high-frequency channel must be consistent with the estimates from the low-frequency channel at the corresponding image location.

A simple way of enforcing the spectral continuity principle is by a "coarse-to-fine" control strategy. In this strategy, the processing proceeds from the low- to the high-frequency channels. The displacement estimate for a pixel in a low-frequency channel determines the center of the search area for the correspoding pixels in the next higher frequency channel. The scale-invariance property of the measurement process suggests that the radius of the search areas between two adjacent channels should be proportional to the scale factor in order to ensure scale invariance of the computations. Once again, note that such scaling is automatically achieved in the pyramid representation.

2.5 The confidence measure

In general, there will be areas of the image with insufficient information at a particular scale for the local determination of displacements. Therefore a confidence measure should be computed along with each match at each scale to indicate whether or not to accept that match for further processing.

Since the image displacement is a vector quantity, it is possible that different directional components of the displacements may be locally computable with different degrees of reliability. For instance, it is intuitively clear that in a homogeneous area of the image no component of the displacement can be reliably estimated. At a point along a line (or an edge), the component perpendicular to the line can be reliably computed, while the component parallel to the line may be ambiguous. Finally, at a point of high curvature along an image-contour it may be possible to completely and reliably determine the displacement vector based on local information. These observations suggest that the confidence measure should be directionally selective, i.e., that it should associate different confidences with the different directional components of the displacement vector. In addition, while an area may be homogeneous at one scale, it may have information useful for reliable matching at a different scale. Hence, the confidence measures should be separately computed within each spatial-frequency channel.

2.6 Smoothness constraint

The computation of a dense displacement field may require "filling in" areas with unreliable displacements based on the reliable displacements in their neighborhood. Such a filling in process can be based on the assumption that the displacement field varies smoothly over the image area covered by a single surface.

The smoothness assumption is violated at locations of discontinuities in the image motion. Such discontinuities arise at surface boundaries of a single object, or at object boundaries due to independent movement of two different objects. Any scheme that uses the smoothness assumption should also include mechanisms for detecting and processing such violations. While some techniques (e.g., see [6]) have methods for processing the known violations of the smoothness constraint, the detection of discontinuities in image motion remains a major unsolved problem.

The most common use of the smoothness assumption can be found in gradient-based techniques for determining image-velocities. In these techniques, a smoothness constraint is formulated as a variational problem involving the minimization of an error associated with a velocity field. In our framework, we suggest the use of a similar smoothness constraint within each channel. After the displacements and the associated confidences are computed within each channel, the displacement field should be smoothed before it is projected to the next higher frequency channel. During the smoothing process, the confidence measures should be used to retain the reliable displacements while allowing the less reliable estimates to change.

3 The Major Approaches in Computer Vision for the Measurement of Motion

As noted in the introductory section, the current computer vision techniques for the measurement of motion can be divided into the gradient-based techniques and matching techniques. In this section, the general principles underlying the computations each of these two categories of techniques are described. The two hierarchical gradient-based algorithms of Enkelmann and Glazer and the hierarchical matching algorithm of Anandan are outlined. The detailed reviews of several other techniques that belong to each category can be found in [7].

3.1 The gradient-based approaches

Almost all the techniques for measuring the instantaneous image-velocities use the gradient-based approach. This approach is based on the assumption that the intensity of light reflected by a point on an environmental surface and recorded in the image remains constant during a short time interval, although the location of the image of that point

may change due to motion. This assumption leads to the following equation, which is called the *intensity constraint*:

$$|\nabla I|\dot{U}^{\perp} = -I_t, \tag{1}$$

where $|\nabla I|$ is the magnitude of the intensity gradient-vector $\vec{\nabla} I = (I_x, I_y)$, I_t is the temporal derivative of the intensity function, and $\dot{U}^{\perp}$ is the component of the image velocity[1] $\vec{\dot{U}}$ parallel to $\vec{\nabla} I$. The other component $\dot{U}^T$, along the direction perpendicular to $\vec{\nabla} I$, is unspecified by this constraint. Since the orientation of the intensity gradient vector is normal to the direction of the "edge" at a point, $U^{\perp}$ and U^T are respectively called the "normal-flow" and the "tangential-flow" components of the edge. The lack of information regarding the tangential-flow component is known as the "aperture problem".

Since the intensity constraint specifies only one component of the image-velocity at a point, an additional constraint is necessary to completely determine the velocity vector, usually given in the form of a smoothness constraint on the velocity field [20,19,27,29]. For this paper, Horn and Schunk's formulation [20] and Nagel's formulation [27,29] are of particular interest, because they are respectively used in the multi-resolution gradient-based algorithms of Glazer and Enkelmann. These two formulations of the smoothness constraints are discussed in detail within the descriptions of the hierarchical techniques given below. It should be noted that both techniques apply the intensity constraint for the computation of displacement. Such an approximation of velocities by displacements is reasonable because of the hierarchical processing schemes used, in which all displacement measurements are small compared to the scale of image-intensity variations.

Enkelmann's approach

Enkelmann [12] uses the *low-pass pyramid* transform described by Crowley and Stern [11] to create a set of Gaussian low-pass filters. After the construction of a low-pass pyramid from each image, the processing begins at a particular coarse level; the description of his technique does not specify how this level is chosen. At the coarsest level, the initial displacement field consists of vectors of zero length. At all other levels the initial displacement field is determined by projecting the field computed at the adjacent coarser level. The projection process involves a bilinear interpolation of the displacement vectors in a small neighborhood of the field at the coarse level.

Within each level, the process of refining the initial displacements is based on Nagel's gradient-based approach [27,29]. In this approach, an area around each pixel in the image is shifted according to the initial displacement vector at that pixel. The refinement to the initial displacement field is computed by minimizing the functional $E = E_{int} + \alpha^2 E_{sm}$ where E_{int} is a formulation of the intensity constraint mentioned above, and E_{sm} represents the smoothness assumption, and measures the spatial variation of the displacement field. Mathematically,

$$E_{int} = \iint dx\,dy\,\left(\hat{I}(x,y) - J(x+u, y+v)\right)^2 \tag{2}$$

and

$$E_{sm} = \iint dx\,dy\,trace\left[(\nabla U^T)^T W (\nabla U^T)\right], \tag{3}$$

where $\hat{I}$ is the intensity function of the shifted first image, J is the intensity function of the second image, and W is a weight matrix which depends on the spatial derivatives of the image intensity function I. The inclusion of W has the effect of allowing the free propagation of the smoothness assumption along image-contours, while restricting its propagation across the contours.

By using the Euler-Lagrange equations, and ignoring the second-order terms of (u, v), as well as the third and higher order spatial-derivatives of the intensity function, the functional minimization problem is transformed to that of solving the following differential equations:

$$AU + b - \alpha^2 \begin{bmatrix} trace\{W\nabla\nabla u\} \\ trace\{W\nabla\nabla v\} \end{bmatrix} = 0, \tag{4}$$

where

$$A = (\nabla I)(\nabla I)^T + \bar{x^2}(\nabla\nabla I)(\nabla\nabla I)^T, \tag{5}$$

and

$$b = \Delta I(\nabla I) + \bar{x^2}(\nabla\nabla I)(\nabla\Delta I). \tag{6}$$

The $\nabla\nabla$ operator represents the matrix of second derivatives, e.g.,

$$\nabla\nabla I = \begin{pmatrix} I_{xx} & I_{xy} \\ I_{xy} & I_{yy} \end{pmatrix},$$

and $\bar{x^2}$ denotes the size of a small image-window around a point (x, y) which represents that point. Note that in the limit, when the time interval between the two frames tends to zero, the displacements in the above differential equation can be replaced by the corresponding image-velocities, provided ΔI is replaced by I_t, the temporal intensity derivative.

By using the finite-difference approach, the differential equations are further transformed into a large sparse system of linear equations. These linear equations are then solved using a multi-resolution relaxation approach. The details of the relaxation process are not relevant for the purposes of this paper, although they may be important for an efficient implementation.

Glazer's approach

Glazer also uses a Gaussian low-pass pyramid representation of the input images and employs a hierarchical version of the Horn and Schunck approach. However, the

[1] In computer vision, the convention has been to denote the image-velocity by $\vec{U}$. However, we wish to reserve that symbol for the image-displacements between successive frames. Hence, we have used $\vec{\dot{U}}$ to denote the velocity.

exact algorithm for the construction of the pyramid is different; Glazer uses Burt's Gaussian-pyramid transformation described in [9]. After the construction of the low-pass pyramids, the processing begins at a coarse level at which the magnitudes of the displacements are expected to be less than a pixel. A coarse-to-fine control-strategy is used; the projection of the displacements between adjacent levels is according to the quad-tree connectivity, wherein each pixel at a coarse level is regarded as the "parent" of four pixels at the adjacent finer level. Each "child" uses the displacement of the parent pixel as its initial displacement.

As in Enkelmann's approach, a window around each pixel in the first image is shifted according to the initial displacement at that pixel. The refinement process also consists of minimizing the sum of two functionals E_{int} and E_{sm}, which represent the intensity constraint and the smoothness assumption. Glazer defines the two errors as

$$E_{int} = \iint dx\,dy\,|\nabla I|^2(U^{\perp} - V^{\perp})^2, \qquad (7)$$

and

$$E_{sm} = \iint dx\,dy\,(\nabla U^T)^T(\nabla U^T). \qquad (8)$$

As before, $U^{\perp}$ is the component of the displacement vector $\vec{U}$ parallel to the intensity-gradient vector $\vec{\nabla} I$, and $V^{\perp} = -\Delta I/|\nabla I|$. Once again, replacing ΔI by I_t leads to a similar equation involving the image-velocity $\vec{U}$.

Glazer also uses the Euler-Lagrange equations to transform this problem into a set of differential equations, and obtains a system of linear equations by using the finite-difference approach. The set of differential equations he obtains are,

$$\nabla I\left[(\nabla I)^T U + I_t\right] - \alpha^2 \begin{bmatrix} trace\{\nabla\nabla u\} \\ trace\{\nabla\nabla v\} \end{bmatrix} = 0, \qquad (9)$$

where the operator $\nabla\nabla$ is as defined above.

He also uses a multi-resolution relaxation process to solve his system of equations, although his approach is more complex than Enkelmann's method and is based on recent thoeretical work concerning general multi-level relaxation techniques. It involves dynamic switching between the levels (up and down) according to the current rate of convergence during the course of the process. The hierarchical gradient-based approach and multi-level relaxation are both described in detail in [15].

3.2 The matching approaches

Whereas the gradient-based approaches use the continuous variations of the intensity over time, the matching approaches are formulated for determining the displacements of image-events from two discrete frames. The type of matching approach is characterized by the choice of the "image-event" which is matched and the criterion for matching that event. These techniques can be broadly classified as *correlation-based* matching approaches and *symbolic token matching approaches*.

Correlation-based matching - Correlation-based matching involves representing an image pixel by a template window centered at that pixel. The "match-measure" between two pixels is computed by comparing the intensities at the corresponding positions of the two template windows. The most common match-measures are direct correlation, mean normalized correlation, variance normalized correlation, and the sum-of-squared-differences. In some cases, the match measure may be a *weighted* sum of the individual pixel comparisons. Usually, the weights are chosen to increase the contribution of the pixels. The "best-match" of a pixel is determined by optimizing the match-measure over a set of candidate-match pixels.

Symbolic-token matching - Symbolic matching techniques use a symbolic representation of geometric structures in the image as the basis for matching. Such an approach is motivated by the view that geometric structures in the image often correspond to "interesting" physical structures in the 3D environment. These structures are interesting because they may be easily distinguishable from other such structures and are likely to be stable over several image frames. In practice, however, the extraction of useful symbolic structures is difficult, and attention is focused on interesting image points and edges.

The most successful matching approaches for computing image-displacements use a hierarchical matching approach [6,10,14,22,23,26,38]. In most of these techniques, the hierarchical approach is chosen for the same reasons as those used in the development of our framework. Of these, the one that is most consistent with our framework is Anandan's technique. This technique is described below. For a detailed discussion of the relationship of each of the hiearchical matching techniques to our framework, refer to [7].

Anandan's hierarchical matching technique

Anandan uses Burt's Laplacian pyramid transformation [9] to create a set of difference-of-Gaussian filters. This process involves computing the difference between the images at each pair of adjacent levels of Burt's Gaussian pyramid. The Laplacian pyramid provides a set of *band-pass filters* rather than a set of low-pass filters. The motivation for choosing band-pass filters was based on empirical studies [8] which indicate that correlation matching is more reliable when such filters are used.

As in Glazer's algorithm, the process begins at a coarse level at which the magnitudes of the displacements are less than a pixel. A coarse-to-fine control strategy is also used, although the method of transferring displacements to a finer level is somewhat different from that used by Glazer and Enkelmann. In this method, which is called

the *overlapped pyramid* projection scheme, each child pixel has four parents and can therefore have up to four distinct initial displacements. The search area for the best match consists of the union of all the pixels in the four 3×3 pixel areas surrounding the four initial match estimates. For details, see [5,7].

The match-criterion used is the minimization of Gaussian-weighted sum-of-squared-differences (SSD) with 5×5 pixel template windows. A search process was used to determine the best match pixel contained in the search area defined above. Recognizing the fact that a match so obtained may not always be unique, Anandan computes a direction-dependent confidence measure associated with the best match.

The confidence measure is derived from the distribution of the SSD values around the best match estimate. Two confidence values C_{max} and C_{min}, which are proportional to the two principal curvatures of the SSD surface are associated with the displacement components along the directions of the two principal axes of that surface. The unit-vectors along the principal axes are denoted as $\hat{e}_{max}$ and $\hat{e}_{min}$. It has been shown that these measures distinguish between corners, edges, and homogeneous image areas [6,7].

A smoothness constraint is formulated as the minimization of the sum of two functionals E_{app}, and E_{sm}. Let $D(x,y)$ denote the displacement obtained for the pixel located at image position (x,y) by minimizing the SSD measure as noted above. The functional $E_{app}(U)$ measures how well a displacement-field $U(x,y)$ approximates $D(x,y)$, while the functional $E_{sm}(U)$ is identical to the one used by Glazer. Mathematically,

$$E_{app} = \iint dx\,dy \left[C_{max} \left((\vec{U} - \vec{D}) \cdot \hat{e}_{max} \right)^2 + C_{min} \left((\vec{U} - \vec{D}) \cdot \hat{e}_{min} \right)^2 \right]. \tag{10}$$

Anandan uses the finite-element approach to solve the minimization problem, which also leads to a relaxation algorithm. The finite-element method is chosen because it allows the inclusion of known discontinuities in the displacement field; the propagation of the smoothness constraint across such discontinuities can be avoided. Several ideas for the detection of discontinuities are proposed, although these ideas have not been tested thoroughly.

4 Relationship of the Hierarchical Approaches to Our Framework

This section explains the relationship of the three hierarchical techniques to our framework. Our purpose here is to show that all the components of the framework are present in each of the three techniques, and explain the particular implementation choices made for those components.

From the descriptions given above, it should be obvious that all techniques use spatial-frequency channels, and a coarse-to-fine control strategy. Glazer and Enkelmann each use Gaussian low-pass filters, whereas Anandan uses band-pass filters. From our point of view, band-pass filters are more appropriate, because they offer a greater separation of the spatial-frequencies. The control-strategy is also similar, except for the differences in the projection scheme.

As noted earlier, Anandan uses the minimization of the SSD as the match-criterion, computes direction-dependent confidence measures, and includes a smoothness constraint which uses the confidence measures. Hence, his technique is completely consistent with our framework. On the other hand, the match-criterion, the confidence measures, and the smoothness constraint have not been made explicit in the two gradient-based techniques. However, a close analysis of the mathematical formulations of the minimization problems reveals that these three components have been implicitly used. In this section, we establish a one-to-one relationship between the minimization problems formulated in each of the gradient-based techniques and in Anandan's matching technique.

In all three cases, the smoothness constraint is formulated as the functional E_{sm}. Although Enkelmann's formulation appears different from the others because of the inclusion of the weight matrix W, the other two formulations can be cast in the same form by using the identity matrix in the place of W.

It has been shown in [7] that a correspondence can be made between the functionals E_{int} used by Glazer and Enkelmann and the E_{app} used by Anandan. This correspondence can be summarized as the following theorem:

Theorem *In the limit, when the inter-frame time interval tends to zero, the formulation of the approximation error for image displacements used in the discrete matching approach converges to the second-order formulation of E_{int} for image-velocities used in the gradient-based approach, provided the third and higher order spatial intensity derivatives are ignored. Further, when the window-size represented by $\bar{x}^2$ tends to zero, E_{app} converges exactly to the first order gradient-based formulation of E_{int}.*

The proof of this theorem is too long for this paper, and can be found in [7]. However, the approach is outlined below, which also identifies the confidence measures implicitly used in the gradient-based formulations.

It has been shown by Nagel [28] that if the third and higher order spatial-derivatives are ignored, the minimization of the SSD measure is equivalent to solving the equation $AU = -b$, where A and b are as defined in equations (5) and (6). This equation is reproduced below:

$$\left((\nabla I)(\nabla I)^T + \bar{x}^2 (\nabla\nabla I)(\nabla\nabla I)^T \right) U = -\left(I_t (\nabla I) + \bar{x}^2 (\nabla\nabla I)(\nabla I_t) \right). \tag{11}$$

In [7], we have also shown that in this limiting case, the principal curvatures and the associated principal axes of the SSD surface are the eigenvalues and the corresponding eigenvectors of A. Since the confidence measures computed by Anandan are proportional to the principal curvatures, his approximation error can be rewritten as

$$E_{app} = \iint dx\,dy \left[\lambda_1 \left((\vec{U} - \vec{D}) \cdot \hat{e}_1\right)^2 + \lambda_2 \left((\vec{U} - \vec{D}) \cdot \hat{e}_2\right)^2\right] \tag{12}$$

where $\vec{D}$ is any solution to equation (11), λ_1 and λ_2 are the two eigenvalues of A, and $\hat{e}_1$ and $\hat{e}_2$ are the associated unit eigenvectors.

Equation (12) can be further rewritten as

$$E_{app} = \iint dxdy(U - D)^T A(U - D). \tag{13}$$

In [7] we also show that the use of Euler-Lagrange equations on the functional E_{app} leads to the differential equation (4). Hence, for all practical purposes, Enkelmann's intensity constraint given in equation (2) is equivalent to the E_{app} used by Anandan.

If in the formulations of A and b, the window size $\bar{x^2}$ is set to zero, equation (11) reduces to

$$(\nabla I)^T U = -I_t,$$

which is equivalent to the normal-flow equation (1). Moreover, it can be shown that in this case,

$$\lambda_1 = 0,\ \lambda_2 = |\nabla I|^2, \text{ and } \hat{e}_1 = \hat{e}_{\nabla I}. \tag{14}$$

With these substituions, the formulation of E_{app} given in equation (12) reduces exactly to that used by Glazer. From this, it is clear that Glazer determines the normal-flow component $V^\perp$ to be $-I_t/|\nabla I|$, and associates a confidence of $|\nabla I|^2$ with it. The tangential-flow component can be arbitrarily set to any value with a confidence measure of zero.

The direction-dependent properties of the second-order equation $AU = -b$, and the behavior of its solution at corners, edges, and homogeneous areas have recently been discussed by Nagel in [30]. However, the fact that the eigenvalues can be regarded as confidence measures of the displacement-components along the directions of the eigenvectors has not been mentioned. In particular, the key equations (13) and (12) have not been derived; instead, in [29] the differential equation $Au = -b$ has been obtained in a more indirect manner. The two major advantages of the explicit identification of the confidence measures are: (i) the three techniques are unified into a single framework, and (ii) the confidence measure can also be used outside the smoothness constraint as in [2].

In summary, the three most successful hierarchical approaches appear to be completely consistent with our framework. Moreover, the three approaches are essentially the same. The distinctions arise in the exact formulations of the various components (e.g., the use of the matrix W in the smoothness constraint, the method of coarse-to-fine projection, etc.), and in the type of input assumed, i.e., continuous image-stream vs. discrete image sequence. Since our framework allows the explicit identification of the various components, the development of a single robust system which incorporates the best implementation choices for each component is within reach.

5 Processing Discontinuities in Image Motion

In this section, we consider one of the major unsolved problems in the analysis of visual motion in relation to our computational framework. This situation involves processing discontinuities in image motion, which are present at the boundaries of surfaces, or at the boundaries of objects. Around the locations of such discontinuities, the smoothing involved in the spatial-frequency decomposition process creates intensity structures that have no physical correlates. Therefore, the information contained in the lower-frequency channels in the two images will be inconsistent. The local translational assumption and the spectral continuity principles are also violated. Finally, it would be inappropriate to apply the smoothness constraint across such boundaries.

The inconsistency between expected behavior of the matching process and the actual behavior are possible clues to the detection of such discontinuities. For instance, a large value for the minimum of the SSD measure could indicate the presence of occlusion or discontinuities in image motion. A comparison of the shape of the auto- and cross- SSD surfaces may also help in the identification of such discontinuities. A detailed discussion of the behavior of the SSD surfaces at occluded areas and at locations of discontinuities in image motion can be found in [7].

6 Demonstrative Results

This section demonstrates the feasibility of the ideas contained in our framework by showing the performance of Anandan's matching algorithm on a pair of real images containing both camera as well as independent object motion. This algorithm has been tested on more than a dozen image pairs, and its performance appears to be fairly consistent [7].

The input images are the two 128×128 resolution images shown in figure 2. The scene consists of a toy dinosaur, a toy chicken in the background, and a tea box in the foreground, all of vhich rest on a table-top which has a grid-pattern on it. The 3-D motion between the two frames consists of a translation of the camera to the right, along with a leftward rotation about the vertical axis (in order to bring the scene back into view), as well as an independent movement of the toy dinosaur. This scene is of interest

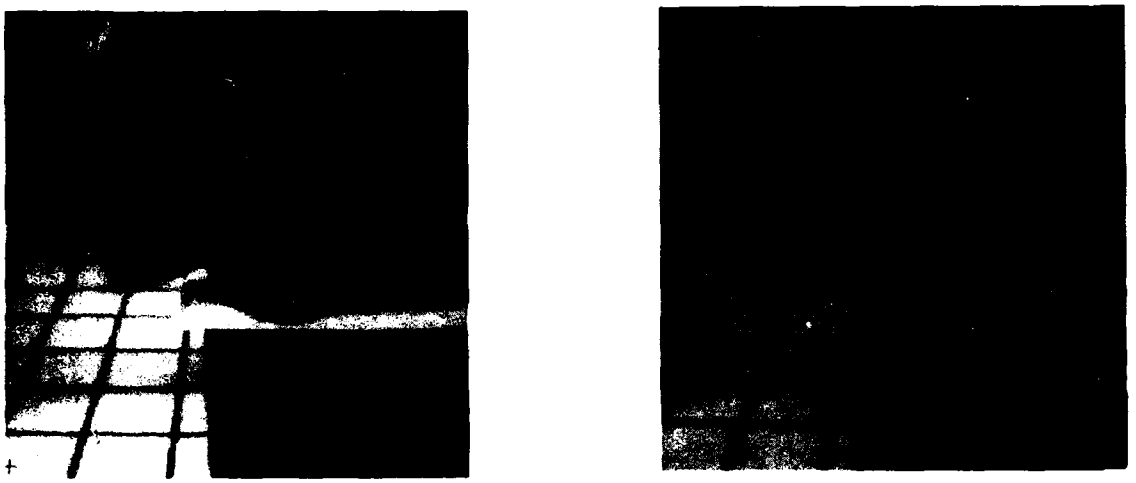


Figure 2: The input images for the dinosaur-image experiment

Figure 3: The un-smoothed displacement vector fields at the finest level of the pyramid. In order to enhance visibility only a 32 × 32 sample of the displacements have been shown.

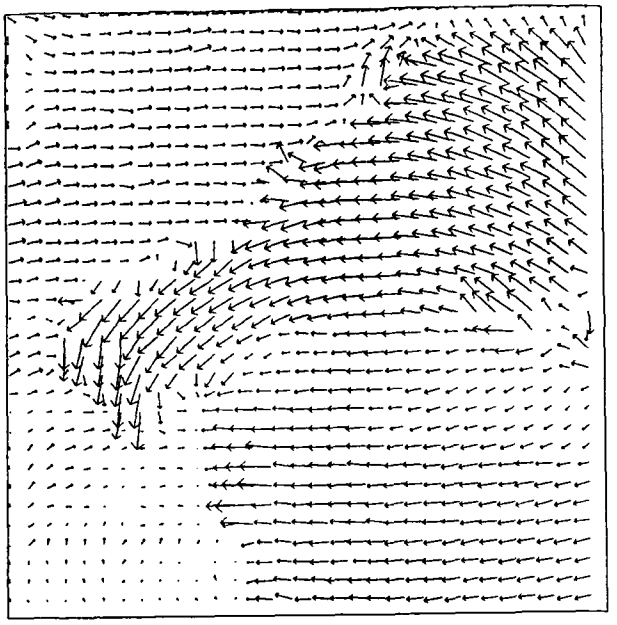


Figure 4: The smoothed displacement vector fields at the finest level of the pyramid. In order to enhance visibility only a 32 × 32 sample of the displacements have been shown.

for the obvious reason that it contains an independently moving object besides a complex camera motion.

Figure 3 shows the displacement field at the finest level of the pyramid computed by the hierarchical matching process *without smoothing*, while Figure 4 displays the displacement field at the finest level produced by the hierarchical algorithm *with smoothing*. Figure 5 displays the smoothed displacement field superimposed on the first frame. Figure 6 displays the confidence measure at the finest level.

It is evident from the figures that the algorithm performs remarkably well on this real image with complex motion. It also appears that the smoothness constraint has generally been useful in "filling in" at several areas of the image, e.g., the lower-right portion of the dinosaur, part of the chicken, and the floor.

The expected behaviors of the confidence measures at corners, edges, and homogeneous areas are confirmed by the displays in Figure 6. In order to make these behaviors more explicit, we attempted to classify the image pixels

according to the following criteria:

- if $c_{min} > 0.5$ the pixel is classified as a corner,
- if $\frac{c_{max}}{c_{min}} > 100$ and $c_{max} > 1$, the pixel is classified as an edge, and
- if $c_{max} < 0.5$ and $\frac{c_{max}}{c_{min}} < 5$ the pixel is classified as a point in a homogeneous area.

Figure 7 displays the results of this classification.

Two areas of the image of particular interest are at the boundary between the chicken and the dinosaur, and the boundary between the tea box and the floor. The first area has been maintained correctly during the smoothing process. This is because the confidence values are rather large for the vectors on either side of this boundary, and this prevents those vectors from changing during the smoothing process. However, the area of the floor just left of the

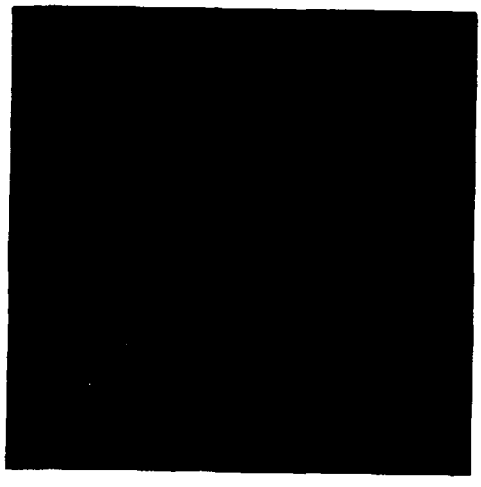


Figure 5: The smoothed displacement vector field at the finest level of the pyramid superimposed on the first input frame. In order to enhance visibility only a 32×32 sample of the displacements have been shown.

Figure 7: The dinosaur-image experiment: The classification of pixels as corners, edges, or homogeneous areas. The top-left quadrant contains the input image. In the images shown in the top-right, bottom-left, and bottom-right quadrants, the corners, edges, and the homogeneous areas have been highlighted.

Figure 6: The confidence measures at the finest level of the pyramid. The confidence c_{max} is displayed on the left as an intensity image, while the confidence c_{min} is displayed on the right. The direction vectors $\hat{e}_{max}$ are superimposed on the c_{max} image.

tea box has displacements that are obviously incorrect, because the highly reliable displacements at the edge of the tea box have influenced their less reliable neighbors.

7 The Modified Hierarchical Framework

7.1 An overview of energy models

Recently, a number of psychophysical models [1,35,36] for the measurement of motion in the human visual system have been shown to be closely related to each other. These have been grouped under the title "spatio-temporal energy models" by Adelson and Bergen [1].

All of the energy models have focused on the problem of determining the direction (and possibly the speed) of motion of a one dimensional signal. Based on the fact that the movement of a 1-D signal with a sharp gradient creates a linear structure in the spatio-temporal domain, the problem of determining the speed of the signal is equated to that of determining the slope of the linear structure. As explained by Adelson and Bergen [1], the determination of this slope involves the construction of a set of simple linear spatio-temporal energy detectors. Each of these detectors can be regarded as a simple linear filter which is tuned to a specific area of the visual field (i.e., an area of the image plane), a specific range of spatial-frequencies, and a specific "sign" of motion (i.e., right, left, or stationary). As it is usually the case with simple linear filters, the response of each unit increases from zero to a peak value within a finite range of locations, frequencies, and speeds. It also appears that an inverse relationship exists between the spatial-frequency and the speed to which a unit is tuned, i.e., units that are tuned to low spatial-frequencies are more sensitive to larger speeds and vice versa.

Since these models have been developed recently, there has not been much effort to apply them to a sequence of two-dimensional images. An effort towards this problem can be found in the current work of Heeger [18], which involves the separation of the two-dimensional image into a set of one-dimensional signals by using filters which are tuned to specific image-orientations. However, the problem of combining the information from the units tuned to different spatial-frequencies, image-locations, and orientations has not yet been fully addressed.

7.2 Using orientation-selective filters

The inverse relationship between the spatial-frequency and the velocity tuning of the energy measurement unit suggests that it may be possible to modify our framework to encompass the energy models. In fact, the relationship between the energy models and the gradient-based approach for velocity computation has been discussed by Adelson and Bergen [1]. It is also possible to cast the SSD minimization process as a measurement of spatio-temporal energy. Based on these relationships, we have recently proposed a modified hierarchical framework which unifies the energy models, the gradient-based approaches and the matching approach. In this section, we provide an outline of this framework.

As illustrated in Figure 8, the new framework consists of a set of "motion-detection" units. Each unit is "tuned" to (i) a specific location on the image-plane, (ii) a specific range of spatial-frequencies and an associated range of speeds, (iii) a specific range of image-orientations and an associated range of directions of movement, and (iv) a "sign" of motion (left, right, or stationary). A confidence measure is included with the output of every unit. Depending on whether the input is a continuous image-stream or a discrete set of images, the measurement of spatio-temporal energy, the gradient-based approach, or a matching approach can be used for computing the response of each individual unit.

The output of each unit can itself be regarded as a measurement of motion. One method of combining these outputs in order to obtain the image displacement field may involve using the following set of combination principles: (i) spectral continuity, which was explained before, (ii) spatial coherence, which is a general form of a smoothness-constraint, (iii) orientation consistency, which means that the units tuned to different orientations all measure the same underlying movement, and (iv) temporal coherence, which means that the velocity of a point in the environment varies smoothly over time.

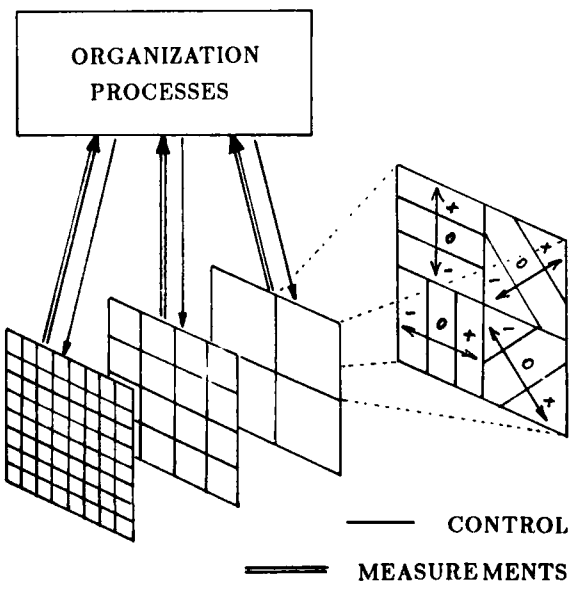


Figure 8: The orientation-selective framework

The motivation for the computation of image flow (i.e., dense displacement fields) arises out of the numerous analyses that relate the structure of the image flow field to that of the environmental surfaces and the parameters of motion. In more general terms, however, a dense displacement may not be necessary. Instead, we can consider methods that directly use the outputs of the energy-measurement units for visual perception and image understanding.

The outputs of the localized energy-measurement units can feed directly to various higher-level processes for a diverse range of goals, e.g., navigation, spatial organization, determination of 3-D structure (see Figure 8). In this sense, the measurements can be treated as "feature" values with an associated confidence measure. Equivalently, the output of each of these units can be regarded as a piece of evidence for the presence of an intensity structure at a specific scale, orientation, and velocity with an associated uncertainty measure. There are a number of schemes currently under investigation for the combination of such probabilistic "features". Most of these approaches appear suited for a connectionist model of computation.

We can also allow the measurement processes to be actively controlled by the high-level processes. Although the units themselves are simple and perform simple convolutions and maximum selection operations, by controlling their input they can be used to achieve the more complex goals of the higher-level processes.

As an example of the use of the energy measurement units, consider the spatial organization process. First, a rough separation of rapidly moving areas from stationary areas can be obtained from the low-frequency tuned units. The low-frequency units near the boundaries of motion will have low-confidence information, and should therefore be ignored. The coarse-to-fine strategy can be applied selectively to the units in the moving area to obtain high-frequency measurements of movement, while simultaneously refining the segmentation. As the refinement of the segmentation progresses, closer attention can be paid to the boundary units, whose search areas may be redefined according to the motion of the area containing them. As much as possible, the process of spatial organization (or "grouping") should progress without a 3-D interpretation, thereby avoiding the various numerical instabilities encountered in the interpretation process. Once the grouping and segmentation is finished, the high resolution processes can provide highly accurate measurements of motion, which can then be used for the determination of the 3-D structure and the 3-D motion parameters.

Any scheme for the use of these energy measurement units should also include an understanding of how these models behave under a tracking situation, i.e., when the camera (or the eye) is rotated to fixate the image of a particular environmental point on the image-plane (or the retina). Here, it is not sufficient to consider the output of the measurement-units under tracking. A comprehensive analysis must also address methods of using the measurements to trigger the tracking mechanisms and schemes for the incremental development of a 3-D model of the environment.

Since this framework is new, no algorithm completely consistent with it exists at present, although Heeger's effort [18] towards using the energy models for two-dimensional images appears to be a step in that direction. In our own future research, we intend to further investigate this problem by identifying the goals of the organization process, and by understanding the mathematical relationships of the 3-D motion parameters and the geometry of the environmental surfaces to outputs of the motion-detectors.

8 Summary

We have described a hierarchical computational framework for the determination of dense displacement fields from a pair of images. The framework is sufficiently general in order to unify the gradient-based and the matching techniques. In particular, we have shown that three successful current techniques [6,15,12] are consistent with our framework. We have also shown that in the limit the approximation error used in the matching technique converges to the intensity constraint used in the gradient-based techniques.

We have also proposed a novel connectionist framework for the measurement of motion which unifies the energy models, the gradient-based approaches, and the matching approach. A comprehensive examination of alternate approaches for the analysis of visual information which use our new framework is a significant research effort in itself and can form the basis of future work in this area of Computer Vision.

ACKNOWLEDGEMENTS

The author wishes to thank Profs. Edward Riseman and Allen Hanson for their continued advice and support through the course of this work. Thanks are also due to Dr. George Reynolds, Prof. Riseman, and Michael Boldt for their comments on earlier drafts, to Dr. Richard Weiss for his help with some of the mathematical aspects of this research, and Dr. Mark Snyder for teaching me some fundamental concepts in linear algebra. Finally, thanks are due to the members of the UMass VISIONS group for creating a unique and valuable research environment.

References

[1] Adelson E. H. and Bergen J. R. Spatiotemproal energy models for the perception of motion, *J. Opt. Soc. Am. A* Vol. 2, No. 2, pp. 284-299, 1985.

[2] Adiv G, Determining 3-d motion and structure from optical flows generated by several moving objects, *IEEE T-PAMI*, 7 (4), pp. 384-401, 1985.

[3] Adiv G., Inherent ambiguities in recovering 3-d motion and structure from a noisy flow field, *Proc. CVPR*, pp. 70-77, 1985.

[4] Aggarwal J. K., Structure and motion from images: fact and fiction, *Proc. of the third Workshop on Computer Vision: Representation and Control*, pp. 127-128, 1985.

[5] Anandan P., Computing dense displacement fields with confidence measures in scenes containing occlusion, *SPIE Intelligent Robots and Computer Vision Conference*, Vol. 521, pp 184-194, 1984, also *COINS Technical Report 84-32*, University of Massachusetts, December 1984.

[6] Anandan P. and Weiss R., Introducing a smoothness constraint in a matching approach for the computation of displacement fields, *DARPA IU Workshop Proc.*, pp. 186-196, 1985.

[7] Anandan P., Measuring visual motion from image sequences, *Ph.D. dissertation*, COINS Department, University of Massachusetts, Amherst, Mass., *in preparation*.

[8] Burt P. J., Yen C. and Xu X., Local correlation measures for motion analysis: A comparative study, *IEEE Proc. PRIP*, 269-274, 1982.

[9] Burt P. J., Fast filter transforms for image processing, *Computer graphics and image processing*, 16, pp. 20-51, 1981.

[10] Burt P. J., Yen C. and Xu X., Multi-resolution flow-through motion analysis, *IEEE CVPR Conference Proceedings*, June 1983, pp. 246-252.

[11] Crowley J. L., and Stern R. M., Fast computations of the difference of low-pass transform, *IEEE Transactions on PAMI*, vol. PAMI-6, pp. 212-222, 1984.

[12] Enkelmann, W., Investigations of multigrid algorithms for the estimation of optical flow fields in image sequences, *Proc. of the Workshop on Motion: Representation and Control*, S. Carolina, pp. 81-87, 1986.

[13] Fang J. and Huang T. S., Some experiments on estimating the 3-d motion parameters of a rigid body from two consecutive Image Frames, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-6, NO. 5, pp 545-554, September, 1984.

[14] Glazer F., Reynolds G. and Anandan P., Scene matching by hierarchical correlation, *IEEE CVPR conference*, June 1983, pp. 432-441.

[15] Glazer F., Hierarchical motion detection, *Ph.D. dissertation*, COINS Department, University of Massachusetts, Amherst, Ma., February 1987.

[16] Grimson W. E. L., Computational experiments with a feature based stereo algorithm, *IEEE transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-7, No. 1, pp. 17-34, 1985.

[17] Hanson A.. R and Riseman E. M., Processing cones: A computational struture for image analysis, in: *Structured computer vision* Tanimoto S. and Klinger A. (Eds.), Academic Press, New York, 1980.

[18] Heeger D., and Pentalnd A., Seeing structure through chaos, *Proc. of the Workshop on motion: Representation and Control*, S. Carolina, pp. 131-136, 1986.

[19] Hildreth E. C., The measurement of visual motion, *Ph.D. dissertation*, Dept. of Electrical Engineering and Computer Science, MIT, Cambridge, Ma., 1983.

[20] Horn B. K. P., and Schunck B. G., Determining Optical Flow, *Artificial Intelligence*, vol. 17, pp. 185-203.

[21] Klinger A., and Dyer R. D., Experiments on picture respresentations using regular decomposition, *Computer graphics and image processing*, vol 5, no. 1, pp. 68-105, 1976.

[22] Lucas B. D., and Kanade T., An iterative image registration technique with an application to stereo vision, *Proc. 7th IJCAI*, Vancouver, B. C., Canada, pp. 674-679, 1981.

[23] Marr D. and Poggio T., A computational theory of human stereo vision, *Proc. Roy. Soc. London, Ser. B*, 204, pp 301-308, 1979.

[24] Mayhew J. E. W. and Frisby J. P., Psychophysical and computational studies towards a theory of human stereopsis, *Artificial Intelligence*, Vol. 17, pp. 349-385, 1981.

[25] Miller, R. and Q. F. Stout, Geometric algorithms for digitized pictures on a mesh-connected computer. *IEEE T-PAMI*, vol. PAMI-7, pp. 216-228, 1985.

[26] Moravec H. *Robot rover visual navigation*, UMI Research press, Ann Arbor, Michigan, 1981.

[27] Nagel H. H., Constraints for the estimation of displacement vector fields from image sequences, *Proc. IJCAI*, Karlsruhe / FRG, pp. 945-951, 1983.

[28] Nagel H. H., Displacement vectors derived from second order intensity variations in image sequences, *CVGIP*, vol. 21, pp. 85-117, 1983.

[29] Nagel H. H. and Enkelmann W., An investigation of smoothness constraints for the estimation of displacement vector fields from image sequnces, *IEEE transactions on PAMI*, vol. PAMI-8, pp. 565-593, 1986.

[30] Nagel H. H., Image sequences – Ten (Octal) years – from phenomenology towards a theoretical foundation, *Proc. of Eighth ICPR*, Paris, France, 1986.

[31] Rieger J. H. and Lawton D. T., Determining the instantaneous axis of translation from optic flow generated by arbitrary sensor motion, *Proc. ACM Siggraph/Sigart Interdisciplinary Workshop on Motion*, Toronto, pp. 33-41, 1983.

[32] Tanimato S. L. and Pavlidis T., A hierarchical data structure for picture processing, *CGIP*, vol. 4, no. 2, pp. 104-119, 1975.

[33] Terzopoulos D., Mutiresolution Computation of Visible-Surface Representations, *Phd Dissertation*, Massachusetts Institute of Technology, Jan. 1984.

[34] Terzopolous D., Image analysis using multiresolution methods, *IEEE transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-8, No. 2, pp. 129-139, 1986.

[35] van Santen J. P. H. and Sperling G., Elaborated Reichhardt detectors, *J. of Opt. Soc. Am.*, vol. 2, no. 7, pp. 300-321, 1985.

[36] Watson A.B. and Ahmuda A. J., Model of human visual-motion sensing, *J. of Opt. Soc. Am.*, vol. 2, no. 7, 1985.

[37] Waxman A., An image-flow paradigm, *Proc. Workshop on computer vision*, Annapolis, MD, pp. 49-57, 1984.

[38] Wong R. Y. and Hall E. L., Sequential hierarchical scene matching, *IEEE transactions on Computers*, Vol. 27, No. 4, pp. 359-366, 1978.

Hierarchical Gradient-Based Motion Detection

Frank Glazer

Computer and Information Science Department
University of Massachusetts
Amherst, Massachusetts 01003

Abstract

A major class of techniques for the computation of image motion is that of gradient-based methods. The application of gradient-based methods is limited to cases of small motions (a few pixels).

In this report we demonstrate that gradient-based methods may be extended to the computation of large disparities by formulating a hierarchical generalization of the single level method that operates on low-pass image pyramids. *A thorough formal analysis of the gradient-based updating equations is presented as well as the implementation of the method in a hierarchical processing cone algorithm. Experiments are presented both to show how single level methods fail and how the hierarchical method succeeds.*

1 Introduction

Motion analysis can be used in machine vision systems to determine the location, movement, and identity of objects. The first stage in such analysis is the computation of image motion — the motion of components of a dynamic image sequence. One major class of techniques for the computation of image motion is that of *gradient-based* methods [Fennema & Thompson 79, Glazer 81, Horn & Schunck 81]. These methods are characterized by (1) the approximate representation of local image neighborhoods by low order polynomials; and (2) the derivation of constraints on the possible image motions in terms of gradients of the image function, both spatial and temporal. Image motions are determined by combining constraints from multiple image locations.

This report describes a hierarchical approach to gradient-based motion computation. It generalizes gradient-based techniques to handle cases for which the disparities between points in two images are more than a few pixels in length. The hierarchical control is very similar to that used for hierarchical correlation matching [Glazer *et al.* 83, Anandan 84, Glazer 87]. Coarse estimates of disparity at high levels of the processing cone are used at lower levels to guide gradient-based algorithms which compute increasingly accurate (higher resolution) disparity vectors.

This report describes one part of the author's thesis on hierarchical motion detection [Glazer 87]. The other major topics addressed there are (1) fast construction of image pyramids; (2) hierarchical correlation-based motion detection; and (3) multilevel relaxation for optic flow computation. In particular, regarding the first topic, a family of *discrete Gaussian* low-pass filters for building low-pass pyramids is presented that provides good anti-aliasing characteristics, efficient computation, and a good hierarchical Gaussian approximation (in the sense of Burt's hierarchical filters [Burt 81]). These filters are used to build the low-pass pyramids used in the experiments presented in this report. All of the motion computation methods presented in [Glazer 87], including hierarchical gradient-based algorithms, are designed for a hierarchical processing architecture consisting of the *processing cone*—a general computational architecture which is regular, parallel and hierarchical [Hanson & Riseman 74, Hanson & Riseman 80]; and *image pyramids*—multiresolution image representations [Tanimoto & Pavlidis 75, Burt 82].

The success of the gradient-based methods depends on the assumption that the image value (intensity) changes in local neighborhoods can be adequately approximated by the low order terms of their Taylor series expansions. In particular, first order methods depend on the degree to which image structure is locally linear. This assumption is not the case when (1) the optic flow (disparity) is relatively large, or (2) the extent over which the intensity varies

Research supported in part by: DARPA grant N00014-82-K-0464

Author's current address: Digital Equipment Corporation, 77 Reed Rd. (HLO2-3/M8), Hudson, MA 01749

linearly is small. In either case, experiments presented later in this chapter show that a gradient-based algorithm operating at a single level of resolution will fail to detect correct disparities. Figures 8 and 9 show a progression of good to bad results for a sequence of image pairs with increasing disparities.

A little more formally, consider the following first order approximation used in first order gradient-based methods

$$F(x+U, y+V) = F(x,y) + \nabla F \cdot (U,V) \qquad (1)$$

Let r be the radius about (x,y) within which this approximation holds (with some error tolerance ϵ). To use this equation in the computation of a disparity vector (U,V), we need $\|(U,V)\| < r$. This condition may be violated in two ways: (1) $\|(U,V)\|$ is too large,i.e. large disparities; or (2) r is too small: strong second order (or higher) image structure.

The first problem can be overcome if a coarse estimate of disparity is known and only an update need be computed. If the coarse disparity estimate is a good one then the required update vector is small and will lie in the "linear" neighborhood. This suggests a coarse-to-fine approach to disparity computation. The second problem is avoided by eliminating higher-order image structure. Such elimination of fast variations in the image can be done by low pass filtering. Of course this smoothing will eliminate detail in the frames and lower the resolution of the computed disparity field. The accuracy of disparity values is reduced and fast variations are lost. In a sense the disparity has itself been smoothed. This suggests that disparity computations on smoothed data can be performed on subsampled image data.

The combination of coarse-to-fine stages with subsampling at the coarser stages leads us to conclude that *first order gradient-based methods may be extended to non-trivial disparity computations by formulating a hierarchical generalization of the single level method that operates on low-pass image pyramids.* "First order" gradient-based methods are those that only involve first order partial derivatives of the image. (By "non-trivial" we mean disparities more than just a few pixels in length.)

In the hierarchical method, approximate disparities are refined at each level of the processing cone in a coarse-to-fine sequence. At each level, the update vectors are expected to be of bounded magnitude, relative to the image scale (pixel spacing) at that level. This constraint provides a *consistency check* which can be used to "filter" out bad update vectors that might result from noisy image data or violations of the method's assumptions.

The next section summarizes the hierarchical method of applying gradient-based disparity computation. Section 3 presents the technical details. Then, in Section 4, a hierarchical gradient-based algorithm is briefly described. Experiments in Section 5 show how single level methods fail and how the hierarchical method succeeds.

2 The Hierarchical Method

Initially we are given two input images $F_1(i,j)$ and $F_2(i,j)$ for which the disparity between them must be computed. The disparity will be represented as a vector field $(s,t) = (s(i,j), t(i,j))$ such that a given disparity vector $(s(i,j), t(i,j))$ describes the displacement between a point (i,j) in F_1 and its "match" at $(i+s, j+t)$ in F_2. Low pass image pyramids $\mathcal{F}_1$ and $\mathcal{F}_2$ are formed for each of the two frames [Glazer 87, Chapter 2]. Processing begins at a level l_{top}, high enough in the pyramid so that the maximum disparity is less than 1 pixel in magnitude. We assume that the coarse disparity field at level $l_{top} - 1$ is $(\mathbf{0},\mathbf{0})$, that is $(0,0)$ at all pixels.

The following processing is performed in a coarse-to-fine succession of stages, beginning at level l_{top} and proceeding down the pyramid (increasing level numbers) to the finest level. First, the coarse disparity field at level $k-1$ is projected down to level k, giving an **approximate disparity field** at that level. Then using the two frames $f_1 = \mathcal{F}_1^k$ and $f_2 = \mathcal{F}_2^k$ at level k in the image pyramids and the approximate disparity field $(U,V) = (U(i,j), V(i,j))$, an update vector field $(u,v) = (u(i,j), v(i,j))$ is computed. The **update vector field** is added to the approximate disparity field to give the more accurate updated disparity field $(U+u, V+v)$. The update vector field is computed at each point (i,j) by comparing data in the neighborhood of $f_1(i,j)$ to data in the corresponding neighborhood $f_2(i+U, j+V)$. The update vector field (u,v) is the relative translation between these two neighborhoods. It can be computed using a gradient-based method. The relationship between these vectors at a given pixel is shown in Figure 1.

The gradient-based algorithm uses two stages. In the first stage the edge flow is computed at each pixel. It is the component of the update vector parallel to the mean spatial gradient at $f_1(i,j)$ and $f_2(i+U, j+V)$. The edge flow vector, along with the approximate disparity, defines a *constraint line* which is a locus of points upon which the update vector lies. In Figure 1 the edge flow vector is shown as (u_0, v_0).

Various methods of computing optic flow from edge flow are reviewed in [Glazer 87, Chapter 3]. These include: global histogramming using a modified Hough transform [Fennema & Thompson 79]; computing the "pseudo-intersection" (a least square error fit) of a set of constraint lines [Glazer 81]; a global optimization method that computes a smooth motion field which satifies the gradient-based constraints at each pixel [Horn & Schunck 81, Yachida 81, Cornelius & Kanade 83, Nagel 83]; and constraint line clustering [Schunck 83]. We will use an *optimization* method. The next section presents a general development of such a method for use in a hierarchical scheme that utilizes approximate disparity information.

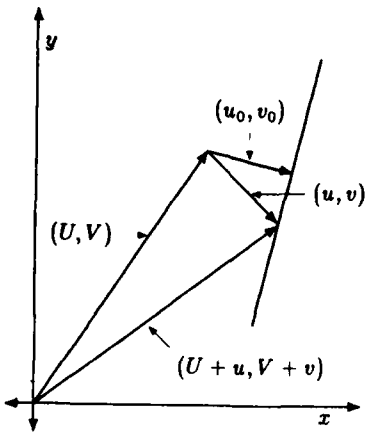


Figure 1: Geometry of updating process

The refinement of the disparity estimate at a given point in the first frame involves the following vectors: (U,V) is the approximate disparity vector; (u,v) is an update vector; and $(U+u, V+v)$ is the updated disparity. The edge flow (u_0, v_0) is the component of (u,v) parallel to the mean spatial gradient of the images at the points being compared. It defines a constraint line which is the locus of points upon which (u,v) is found.

3 Formal Development

3.1 Computing a Refined Disparity Field

Let $F_1(x,y)$ and $F_2(x,y)$ be two frames of an image sequence and let $(U,V) = (U(x,y), V(x,y))$ be a vector field approximately describing the disparity from F_1 to F_2. We want to find a vector field $(u,v) = (u(x,y), v(x,y))$ which "updates" (U,V), i.e. the new vector field $(U+u, V+v)$ is a better approximation of the disparity from F_1 to F_2. In the following development we will use the notation $\mathbf{x} = (x,y)$, $\mathbf{U} = (U,V)$ and $\mathbf{u} = (u,v)$.

3.1.1 The Constraint Line

First we will find a constraint line for $\mathbf{u}$. We define the change in image value between a point $\mathbf{x}$ in F_1 and a corresponding point in F_2 at the displaced point $\mathbf{x} + \mathbf{U} + \mathbf{u}$ as:

$$\triangle F(\mathbf{x}) = F_2(\mathbf{x}+\mathbf{U}+\mathbf{u}) - F_1(\mathbf{x}) \tag{2}$$

Consider the following two ways in which a Taylor series expansion of F_2 can be computed:

$$\begin{aligned} F_2(\mathbf{x}+\mathbf{U}+\mathbf{u}) &= F_2(\mathbf{x}+\mathbf{U}) + \mathbf{u}\cdot G_2(\mathbf{x}+\mathbf{U}) \\ &\quad + \frac{1}{2}\mathbf{u}^T\big(H_2(\mathbf{x}+\mathbf{U})\big)\mathbf{u} \\ &\quad + \text{higher order terms} \end{aligned} \tag{3}$$

and

$$\begin{aligned} &F_2(\mathbf{x}+\mathbf{U}) \\ &= F_2(\mathbf{x}+\mathbf{U}+\mathbf{u}) + (-\mathbf{u})\cdot G_2(\mathbf{x}+\mathbf{U}+\mathbf{u}) \\ &\quad + \frac{1}{2}(-\mathbf{u})^T\big(H_2(\mathbf{x}+\mathbf{U}+\mathbf{u})\big)(-\mathbf{u}) \\ &\quad + \text{higher order terms} \end{aligned} \tag{4}$$

where G_2 is the gradient of F_2 and H_2 is the Hessian of F_2. The gradient and the Hessian of a two dimensional image are generalizations of the first and second derivative of function of one variable. The last two equations can be combined to give:

$$\begin{aligned} &F_2(\mathbf{x}+\mathbf{U}+\mathbf{u}) \\ &= F_2(\mathbf{x}+\mathbf{U}) + \mathbf{u}\cdot\frac{1}{2}\big[G_2(\mathbf{x}+\mathbf{U}) + G_2(\mathbf{x}+\mathbf{U}+\mathbf{u})\big] \\ &\quad + \frac{1}{2}\mathbf{u}^T\frac{1}{2}\big[H_2(\mathbf{x}+\mathbf{U}) - H_2(\mathbf{x}+\mathbf{U}+\mathbf{u})\big]\mathbf{u} \\ &\quad + \text{third and higher order terms} \end{aligned} \tag{5}$$

If $\mathbf{U}+\mathbf{u}$ varies slowly, i.e. $|\nabla(\mathbf{U}+\mathbf{u})|^2$ is small, then the following approximations can be made: $G_2(\mathbf{x}+\mathbf{U}+\mathbf{u}) = G_1(\mathbf{x})$ and $H_2(\mathbf{x}+\mathbf{U}+\mathbf{u}) = H_1(\mathbf{x})$ ([Glazer 87, Appendix D]). We then get:

$$\begin{aligned} &F_2(\mathbf{x}+\mathbf{U}+\mathbf{u}) \\ &= F_2(\mathbf{x}+\mathbf{U}) + \mathbf{u}\cdot\frac{1}{2}\big[G_2(\mathbf{x}+\mathbf{U}) + G_1(\mathbf{x})\big] \\ &\quad + \frac{1}{2}\mathbf{u}^T\frac{1}{2}\big[H_2(\mathbf{x}+\mathbf{U}) - H_1(\mathbf{x})\big]\mathbf{u} \\ &\quad + \text{third and higher order terms} \end{aligned} \tag{6}$$

Substituting into Equation 2 and *ignoring higher order terms* finally gives:

$$\triangle F(\mathbf{x}) = F_2(\mathbf{x}+\mathbf{U}) - F_1(\mathbf{x}) + \mathbf{u}\cdot\frac{1}{2}\big[G_2(\mathbf{x}+\mathbf{U}) + G_1(\mathbf{x})\big] \tag{7}$$

If $\mathbf{U}$ is a good estimate, then $\mathbf{u}$ will be small and hence dropping higher order terms leaves us with a good approximation. In fact the second order term is very small due to the fact that $H_2(\mathbf{x}+\mathbf{U})$ is very close to $H_1(\mathbf{x}) \approx H_2(\mathbf{x}+\mathbf{U}+\mathbf{u})$. That is why we used both of the Taylor series expansions in Equations 3 and 4 and not just one of them.

To simplify our equations we define

$$\hat{F}_2(\mathbf{x}) \triangleq F_2(\mathbf{x}+\mathbf{U}) = F_2(\mathbf{x}+\mathbf{U}(\mathbf{x})) \tag{8}$$

and

$$\overline{F} \triangleq \frac{1}{2}(F_1 + \hat{F}_2) \tag{9}$$

Then the gradient of $\overline{F}$ is equal to $\nabla\overline{F} \triangleq (\overline{F}_x, \overline{F}_y) = \frac{1}{2}(G_1 + \hat{G}_2)$ where G_1 is the gradient of F_1 and $\hat{G}_2$ is the gradient of $\hat{F}_2$. We can then write

$$\triangle F(\mathbf{x}) = \hat{F}_2(\mathbf{x}) - F_1(\mathbf{x}) + \mathbf{u}\cdot(\overline{F}_x, \overline{F}_y) \tag{10}$$

If we set ΔF to zero then equation 10 defines a constraint line upon which the update vector u must lie.

Two key assumptions in the above derivation were that (1) $\mathbf{U(x)} + \mathbf{u(x)}$ varies slowly, and (2) second and higher order terms in the Taylor series expansion can be ignored. The first assumption does not hold at motion boundaries. This limitation is the same as that which occurs when gradient-based flow computations are performed. Hence, similar solutions, whatever they might be, can be used.

The (first order) Taylor series approximation only holds within some local neighborhood. If the disparity is greater than the "radius" of this neighborhood then the method doesn't work. The second assumption is warranted because the approximate disparity field $\mathbf{U(x)}$ allows us to combine higher order components from approximately equivalent image neighborhoods. This is a key element of the hierarchical method. Imagine the above derivation was performed for the non-hierarchical case. This is easily done by setting $\mathbf{U} = (0,0)$, that is, eliminate all appearances of U in Equations 2 to 10. The second order terms in Equations 5 and 6 then contain the factors $\left[H_2(\mathbf{x}) - H_2(\mathbf{x}+\mathbf{u})\right] \approx \left[H_2(\mathbf{x}) - H_1(\mathbf{x})\right]$. These terms will NOT be negligible when the actual disparity is large.

Theoretically, the second assumption does not pose a problem for the case of flow computation which is formulated in a continuous three dimensional XYT space. However, in real imaging systems we are actually processing discrete frames taken with a non-trivial Δt and approximating derivatives with finite differences. This introduces a non-infinitesimal jump which may exceed the radius of approximation.

3.1.2 Edge flow

We define **edge flow** as the point on the constraint line lying nearest the origin. The edge flow is then

$$\begin{aligned} \mathbf{u}_0 &\triangleq \frac{-(\hat{F}_2 - F_1)}{|\nabla F|}\frac{\nabla \overline{F}}{|\nabla F|} \\ &= \left(\frac{-\overline{F}_x(\hat{F}_2 - F_1)}{F_x^2 + \overline{F}_y^2}, \frac{-\overline{F}_y(\hat{F}_2 - F_1)}{F_x^2 + F_y^2}\right) \end{aligned} \tag{11}$$

This vector is used as the initial estimate of the update vector u.

Both u and $\mathbf{u}_0$ lie on the constraint line and $\mathbf{u}_0$ is that point on the line nearest the origin. Thus $\mathbf{u}_0$ must be smaller (in magnitude) than u, that is $\|\mathbf{u}_0\| \leq \|\mathbf{u}\|$. In a hierarchical scheme, approximate disparity vectors U insure us that the update vector must be bounded in magnitude. This in turns places a bound on the magnitude of the edge flow vector. In Section 4.4 this fact is used to introduce a consistency check on the computed edge flow. ***This is an added benefit of the hierarchical method.***

3.1.3 Computing the Update: An Optimization Problem

Equation 10 provides a constraint on the value of the update vector at each point in the image space (at a given level). The assumption that the change in image value at the matching points vanishes, i.e. $\Delta F = 0$, specifies a line in "update vector" space upon which the update vector should lie. Due to noise in either image, this is likely to be too strong a constraint. A more practical constraint requires that the update vector lie as close to the line as possible.

In the second stage of a first order method, other information is brought to bear to finally arrive at update vectors at each pixel. In the motion problem we have posed, disparity vector fields can vary across the image, but locally they are almost constant. Thus we must use local disparity information to arrive at a final update value at a given pixel. We will use an optimization method, which while formulated as a global optimization problem, can be represented as a set of local constraints and can be solved by a local, parallel, uniform algorithm. Horn and Schunck first applied such a method to the computation of optic flow from edge flow [Horn & Schunck 81]. The analysis here is similar, using image differences in place of partial derivatives with respect to time and replacing the optic flow field by an approximate disparity field summed with an update vector field.

The variational problem we wish to solve asks for a disparity update field for which (1) the update vector lies as close to the constraint line as possible, and (2) the disparity field is as smooth as possible. Since "close"-ness and "smooth"-ness are measured in different units, a constant of proportionality (α in the equation below) must be included as a parameter to relate these two values.

The problem then posed is: given an approximate disparity field (U,V), and image gradients $\overline{F}_x, \overline{F}_y, \hat{F}_2 - F_1$, find an update vector field (u,v) which minimizes the functional

$$\begin{aligned} \iint &\left[\overline{F}_x u + \overline{F}_y v + (\hat{F}_2 - F_1)\right]^2 \\ &+ \alpha^2\left[|\nabla(U+u)|^2 + |\nabla(V+v)|^2\right] dxdy \end{aligned} \tag{12}$$

The first term in the functional is the square of the change of image value, as it is defined by Equation 10. The second term is a roughness measure of $U+u$ and $V+v$, where ∇ is the gradient operator and α a relative weighting factor. Note that it is the disparity field $(U+u, V+v)$ which is required to be smooth and not just the update field (u,v). This is very important in the implementation algorithm (Section 4), which truncates the disparity vector as it is passed down a level in the cone.

The equivalent PDE system (Euler's equations, [Courant & Hilbert 53, Section IV.3.4]) is

$$\alpha^2\Delta(U+u) - \overline{F}_x^2 u - \overline{F}_x\overline{F}_y v = \overline{F}_x(\hat{F}_2 - F_1) \quad (13a)$$

$$\alpha^2\Delta(V+v) - \overline{F}_x\overline{F}_y u - \overline{F}_y^2 v = \overline{F}_y(\hat{F}_2 - F_1) \quad (13b)$$

These equations involve first and second partial derivatives in the image data and the vector fields. This is purely local information, that is, at a given point the partial derivatives of a scalar or vector field are determined by their values in the immediate neighborhood of the point. Thus the discrete solution to these equations presented in the next section is a local computation.

3.2 Discrete Representation and Computation

Update vectors will be found by solving Equation 13. The edge flow, given in Equation 11, will provide an initial estimate. To do this we must formulate a discrete representation of these equations. Finite difference approximations to the "continuous" derivative operators will be used. Equation 13 then becomes a system of linear equations in the variables $(u_{ij}, v_{ij}) \triangleq (u(i,j), v(i,j))$. These equations are solved using an iterative relaxation scheme which is local, uniform, and parallel— criteria we require to be satisfied for efficient implementation in a cellular/processing-cone architecture [Glazer 87].

3.2.1 Partial Derivative Operators

Equations 11 and 13 will be represented by discrete finite difference methods. We do this with the following first difference operators:

$$\frac{1}{h}\delta_x \triangleq \frac{1}{2h}[-1\ 0\ 1] * \frac{1}{4}[1\ 2\ 1]^T = \frac{1}{8}\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

$$\frac{1}{h}\delta_y \triangleq \frac{1}{4}[1\ 2\ 1] * \frac{1}{2h}[-1\ 0\ 1]^T = \frac{1}{8}\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

These operators were chosen to combine a first central difference in the direction of the partial derivative and smoothing (low-pass filtering) in the perpendicular direction [Glazer 87, Chapter V, Section 3.2.1]. Note that, if we ignore the 1/8 scaling factor, these operators are the Sobel edge detectors used early on in machine vision.

Using these operators, Equation 11 is approximated by

$$(u_0, v_0) = \left(\frac{-\frac{1}{h}\delta_x F(\hat{F}_2 - F_1)}{(\frac{1}{h}\delta_x F)^2 + (\frac{1}{h}\delta_y F)^2}, \frac{-\frac{1}{h}\delta_y F(\hat{F}_2 - F_1)}{(\frac{1}{h}\delta_x F)^2 + (\frac{1}{h}\delta_y F)^2} \right)$$

$$= \left(\frac{-h\delta_x \overline{F}(\hat{F}_2 - F_1)}{\delta_x F^2 + \delta_y F^2}, \frac{-h\delta_y \overline{F}(\hat{F}_2 - F_1)}{\delta_x F^2 + \delta_y F^2} \right) \quad (14)$$

Using the operators in Equation 13, (and multiplying through by h^2) gives the approximations

$$\alpha^2\Delta_1(U+u) - (\delta_x F)^2 u - (\delta_x F)(\delta_y F)v = h\delta_x \overline{F}(\hat{F}_2 - F_1) \quad (15a)$$

$$\alpha^2\Delta_1(V+v) - (\delta_x F)(\delta_y F)u - (\delta_y F)^2 v = h\delta_y \overline{F}(\hat{F}_2 - F_1) \quad (15b)$$

3.2.2 Normalized Coordinate Systems

The **base coordinate system** is the coordinate system in which the inter-pixel spacing (grid spacing) is 1 unit at the **base level**—the level L of the input images. Using this coordinate system, at level k the grid spacing is $h_k = 2^{L-k}$. Equations 14 and 15 involve the grid spacing h. At the lowest level, $h = 1$ and it can be removed from the equations. The h factor can also be removed from consideration at all other levels of the pyramid if we use a normalized coordinate system at each level. In the **normalized coordinate system** at a given level, the distance between two adjacent pixels is 1 unit. Image operators acting at a single level can be computed without using h_k. However, intra-level computations and comparisons must then take into account the difference in the grid spacings as given by the ratio $r \triangleq h_{k-1}/h_k$.

Consider for example the edge flow equations (Equation 14). If (u_0, v_0) is the edge flow in base coordinates and $(\tilde{u}_0, \tilde{v}_0)$ is the edge flow in normalized coordinates then $h(\tilde{u}_0, \tilde{v}_0) = (u_0, v_0)$. Substitution into equation 14 gives

$$(\tilde{u}_0, \tilde{v}_0) = \left(\frac{-\delta_x \overline{F}(\hat{F}_2 - F_1)}{\delta_x \overline{F}^2 + \delta_y \overline{F}^2}, \frac{-\delta_y \overline{F}(\hat{F}_2 - F_1)}{\delta_x \overline{F}^2 + \delta_y \overline{F}^2} \right) \quad (16)$$

Similarly the h factor can be removed from equation 15 when we specify that (U, V) and (u, v) are represented in the normalized coordinate system. We will assume from now on that this specification has been made and so we need not use the tilde ($\tilde{}$) notation.

3.2.3 Relaxation Equations

Equation 15 can be solved using the iterative point-Jacobi method. This involves the iterative application of the following update equations derived from equation 15:

$$u^{k+1} := \hat{u}^k + (\hat{U} - U) \qquad (17a)$$
$$- \frac{\overline{F}_{\delta x}\left[\overline{F}_{\delta x}(\hat{u}^k + \hat{U} - U) + \overline{F}_{\delta y}(\hat{v}^k + \hat{V} - V) + (\hat{F}_2 - F_1)\right]}{(\alpha^2 + \overline{F}_{\delta x}^2 + \overline{F}_{\delta y}^2)}$$

$$v^{k+1} := \hat{v}^k - (\hat{V} - V) \qquad (17b)$$
$$- \frac{\overline{F}_{\delta y}\left[\overline{F}_{\delta x}(\hat{u}^k + \hat{U} - U) + \overline{F}_{\delta y}(\hat{v}^k + \hat{V} - V) + (\hat{F}_2 - F_1)\right]}{(\alpha^2 + \overline{F}_{\delta x}^2 + \overline{F}_{\delta y}^2)}$$

where $(\hat{U}, \hat{V}) = (\hat{U}(i,j), \hat{V}(i,j))$ are local averages of (U, V); (u^k, v^k) is the estimate at the kth iteration; $(\hat{u}^k, \hat{v}^k)$ are local averages of (u^k, v^k); and $\overline{F}_{\delta x} \triangleq \delta_x \overline{F}$ and $\overline{F}_{\delta y} \triangleq \delta_y \overline{F}$ are the normalized first difference operators.

3.3 Geometric Interpretation of Update Equations

The relaxation update equations (17) are better understood in terms of their geometric interpretation. They can be thought of as entailing two steps for each pixel: (1) compute the average disparity in the local neighborhood; (2) project this value towards the constraint line. The second step is accomplished by picking a disparity vector that lies on the line segment joining the average disparity and its projection onto the constraint line. The choice is made closer to the constraint line when we have higher confidence in the measurement of this line, namely when the spatial gradient at this pixel is high. The following paragraphs elaborate on these ideas.

First consider the case for which the approximate disparity $(U, V) = 0$. Equation 17 then simplifies to

$$u^{k+1} := \hat{u}^k - \frac{\overline{F}_{\delta x}\left[\overline{F}_{\delta x}\hat{u}^k + \overline{F}_{\delta y}\hat{v}^k + (F_2 - F_1)\right]}{(\alpha^2 + \overline{F}_{\delta x}^2 + \overline{F}_{\delta y}^2)} \qquad (18a)$$

$$v^{k+1} := \hat{v}^k - \frac{F_{\delta y}\left[F_{\delta x}\hat{u}^k + F_{\delta y}\hat{v}^k + (F_2 - F_1)\right]}{(\alpha^2 + F_{\delta x}^2 + F_{\delta y}^2)} \qquad (18b)$$

or rearranged into a vector notation

$$(u,v)^{k+1} = (\hat{u},\hat{v})^k - \left[\frac{(\hat{u},\hat{v})^k \cdot (F_{\delta x}, F_{\delta y}) + (F_2 - F_1)}{(\alpha^2 + \overline{F}_{\delta x}^2 + \overline{F}_{\delta y}^2)}\right](\overline{F}_{\delta x}, \overline{F}_{\delta y}) \qquad (19)$$

Note that since $(U, V) = 0$, $\hat{F}_2 = F_2$.

The geometry of this simple relaxation updating equation in shown in Figure 2. The average disparity over the local neighborhood of an image point is shown as the point $(\hat{u}, \hat{v})^k$ in (u, v) disparity space. The constraint line is shown with its equation $(u, v) \cdot (\overline{F}_x, \overline{F}_y) + (F_2 - F_1) = 0$. The (perpendicular) projection of $(\hat{u}, \hat{v})^k$ onto the constraint line is shown as $(\hat{u}, \hat{v})^k - (p, q)$ where (p, q) is the vector from that point to $(\hat{u}, \hat{v})^k$. It is easy to verify that:

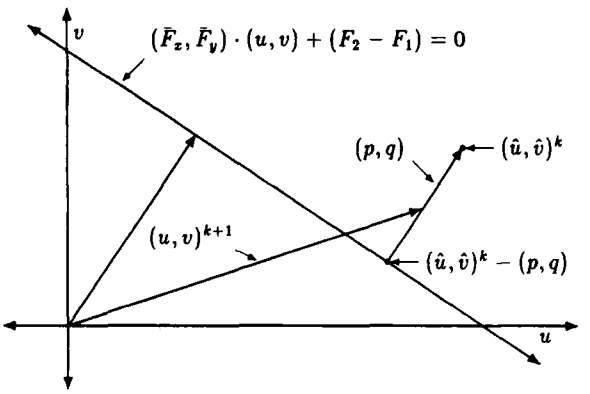


Figure 2: Geometry of simple relaxation updating

$(\hat{u}, \hat{v})^k$ is the average of neighboring pixel flow estimates at iteration k. $(\hat{u}, \hat{v})^k - (p, q)$ is the perpendicular projection of $(\hat{u}, \hat{v})^k$ onto the constraint line. This is the point on the constraint line nearest to $(\hat{u}, \hat{v})^k$. The new flow estimate $(u, v)^{k+1}$ lies between $(\hat{u}, \hat{v})^k$ and $(\hat{u}, \hat{v})^k - (p, q)$.

$$(p, q) = \frac{(\hat{u}, \hat{v})^k \cdot (\overline{F}_x, \overline{F}_y) + (F_2 - F_1)}{F_x^2 + F_y^2}(F_x, F_y) \qquad (20)$$

Now suppose that we were to choose an update vector $(u, v)^{k+1}$ along the line segment joining $(\hat{u}, \hat{v})^k$ and $(\hat{u}, \hat{v})^k - (p, q)$ using a linear combination of these two vectors, that is

$$\begin{aligned}(u,v)^{k+1} &= (1-t)(\hat{u},\hat{v})^k + t[(\hat{u},\hat{v})^k - (p,q)] \\ &= (\hat{u},\hat{v})^k - t(p,q)\end{aligned} \qquad (21)$$

where $t \in [0, 1]$ selects a position between the two end points. When Equation 20 is substituted into this equation, we can make the result equivalent to Equation 19 if we set

$$\begin{aligned}t &= \frac{\overline{F}_x^2 + \overline{F}_y^2}{\alpha^2 + \overline{F}_x^2 + \overline{F}_y^2} = \frac{\|(\overline{F}_x, \overline{F}_y)\|^2}{\alpha^2 + \|(F_x, F_y)\|^2} \\ &= \frac{1}{1 + \left(\frac{\alpha}{\|(F_x, F_y)\|}\right)^2}\end{aligned} \qquad (22)$$

In Figure 3, t is graphed as a function of the magnitude of the spatial gradient $\|\overline{F}_x, \overline{F}_y\|$. We see that when the gradient is small t goes to zero and, looking at Equation 21,

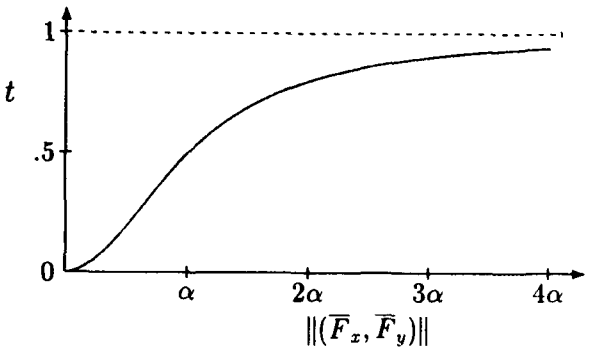


Figure 3: Update weighting factor

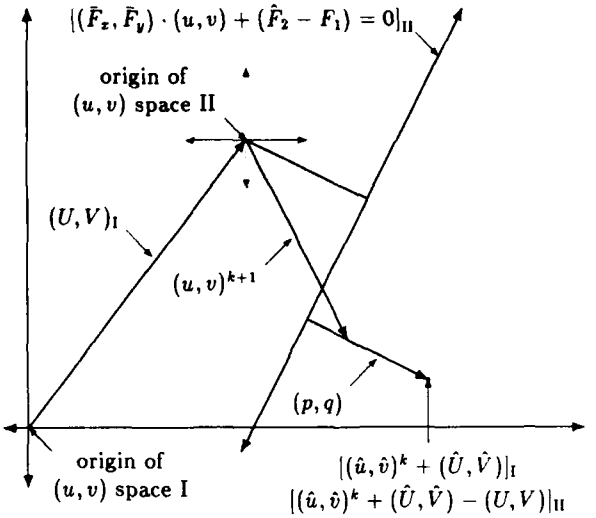


Figure 4: Geometry of hierarchical relaxation updating

$(u,v)^{k+1}$ is chosen near $(\hat{u},\hat{v})^k$. On the other hand as the gradient gets very large t goes to 1.0 and $(u,v)^{k+1}$ is chosen near $(\hat{u},\hat{v})^k - (p,q)$. The midway point, giving equal weight to both possibilities, occurs when the magnitude of the gradient is equal to α. The parameter α controls the shape of this function. Larger values of α "stretch" the curve out to the right so that higher values of the spatial gradient are needed to pull the update vector towards the constraint line.

Now we generalize this picture to the case of hierarchical relaxation updating. Figure 4 shows the geometry of updating for a given location in the image. First note that two coordinate systems are shown in this diagram. The main coordinate system, labeled "I", is the (u,v) space with its origin at the point corresponding to zero disparity. This corresponds to the coordinate space shown in Figure 2. The approximate disparity (U,V) is shown with its tail at the origin of this space. The secondary coordinate system, labeled "II", is the (u,v) space with its origin at the approximate disparity. Subscripts of $_I$ or $_{II}$ are used to specify which space is being used for a given set of coordinates for a point. For example, the approximate disparity can be represented as $(U,V)_I$ or $(0,0)_{II}$ and the updated disparity as $(U + u^{k+1}, V + v^{k+1})_I$ or $(u^{k+1}, v^{k+1})_{II}$. Because the coordinate spaces differ only by a translation, the coordinate representation of a vector (as opposed to a point) is independent of the choice of coordinate space.

The selection of an update vector again involves (1) computation of the average disparity in the local neighborhood, and (2) projection of this value towards the constraint line. The average disparity is the average of the approximate disparity vector summed with the update vector at each pixel in the local neighborhood. This is $(\widehat{u^k + U}, \widehat{v^k + V})_I = |(\hat{u},\hat{v})^k + (\hat{U},\hat{V})|_I$ or in the secondary coordinate system $|(\hat{u},\hat{v})^k + (\hat{U},\hat{V}) - (U,V)|_{II}$. The constraint equation is defined on update vectors (u,v) and hence it is defined in the secondary space. This is signified in the figure by the subscript $_{II}$ given to the equation. Again (p,q) is the vector pointing to the average disparity from its (perpendicular) projection on the constraint line. In this case it is given by

$$(p,q) = \frac{[(\hat{u},\hat{v})^k + (\hat{U},\hat{V}) - (U,V)] \cdot (\overline{F}_x, \overline{F}_y) + (\hat{F}_2 - F_1)}{\overline{F}_x^2 + \overline{F}_y^2} (\overline{F}_x, \overline{F}_y) \tag{23}$$

The generalization to the earlier non-hierarchical case is now straightforward. We choose an update vector $(u,v)^{k+1}$ along the line segment joining $|(\hat{u},\hat{v})^k + (\hat{U},\hat{V}) - (U,V)|_{II}$ and $|(\hat{u},\hat{v})^k + (\hat{U},\hat{V}) - (U,V)|_{II} - (p,q)$ using a linear combination of these two vectors, that is

$$\begin{aligned}(u,v)^{k+1} &= (1-t)|(\hat{u},\hat{v})^k + (\hat{U},\hat{V}) - (U,V)| \\ &\quad + t|(\hat{u},\hat{v})^k + (\hat{U},\hat{V}) - (U,V) - (p,q)| \\ &= |(\hat{u},\hat{v})^k + (\hat{U},\hat{V}) - (U,V)| - t(p,q) \end{aligned} \tag{24}$$

where $t \in [0,1]$ selects a position between the two end points. When Equation 23 is substituted into this equation, we can make the result equivalent to Equation 17 if, as before, we use t as defined in Equation 22. In Figure 3, t is graphed as a function of the magnitude of the spatial gradient $\|F_x, F_y\|$ with the parameter α controlling the shape of this function. When the gradient is small t goes to zero and $(u,v)^{k+1}$ is chosen near $|(\hat{u},\hat{v})^k + (\hat{U},\hat{V}) - (U,V)|_{II}$. On the other hand as the gradient gets very large t goes to 1.0 and $(u,v)^{k+1}$ is chosen near $|(\hat{u},\hat{v})^k + (\hat{U},\hat{V}) - (U,V)|_{II} - (p,q)$. The midway point, giving equal weight to both possibilities, occurs when the magnitude of the gradient is equal to α. Larger values of α "stretch" the curve out to the right so that higher values of the spatial gradient are needed to pull the update vector towards the constraint line.

4 Hierarchical Disparity Algorithms

4.1 Hierarchical Data Flow

Data flow graphs for the hierarchical disparity algorithm are shown in Figures 5–7. Figure 5 shows the coarse-to-fine flow of control and data. The hierarchical disparity computation is a coarse-to-fine algorithm operating on two low-pass image pyramids. The PROJECTION operator passes the updated disparity field to the next level of the processing cone. The REFINEMENT process computes an updated disparity field given an estimated disparity field and the image pyramid data at the corresponding resolution. A data flow diagram of REFINEMENT is shown in Figure 6. Update vectors are computed using an iterative relaxation process shown in Figure 7. A more detailed description of the components of the hierarchical disparity algorithm is given in [Glazer 87, Chapter V, Section 4].

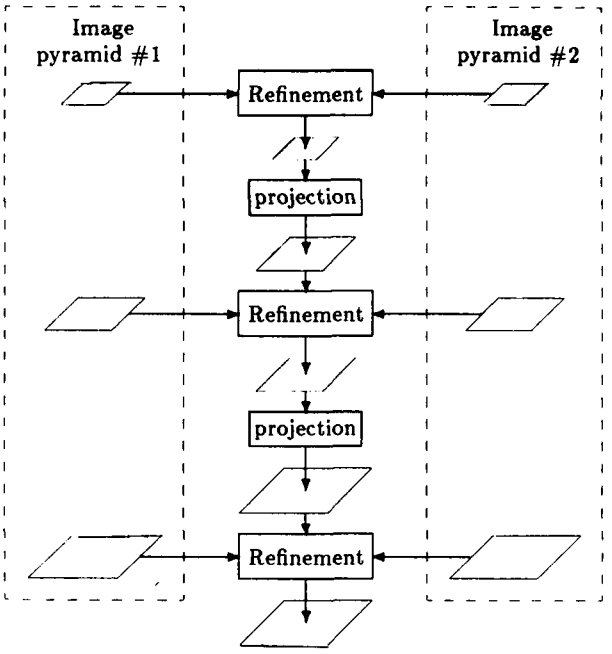


Figure 5: Hierarchical disparity: data flow

The overall hierarchical disparity computation is a coarse-to-fine algorithm operating on two low-pass image pyramids (shown in dotted outline). The REFINEMENT processes compute an updated disparity field given an estimated disparity field and the image pyramid data at the corresponding resolution. The PROJECTION operator passes the updated disparity field to the next level of the processing cone.

4.2 Projection

The Projection operator passes the coarse disparity vector of the father to its four sons with two changes. First, the components of the vector are multiplied by 2, the inter-grid spacing ratio. This converts between the two normalized coordinate systems. Second, the values are then truncated to the nearest integer. This simplifies the computation of gradients to follow.

4.3 Gradients

The gradients that must be computed are $\delta_x F$, $\delta_y F$, and $\hat{F}_2 - F_1$ (see Equations 16–17), where $\hat{F}_2(\mathbf{x}) \triangleq \hat{F}(\mathbf{x}+\mathbf{U})$ and $\hat{F} \triangleq \frac{1}{2}(F_1 + \hat{F}_2)$. Since δ_x and δ_y are linear operators, $\delta_x \hat{F} = \frac{1}{2}(\delta_x F_1 + \delta_x \hat{F}_2)$ and similarly for $\delta_y \hat{F}$. When refinement is being performed at a pixel (i,j), $\delta_x F_1$ is a first difference of F_1 computed at (i,j) and $\delta_x \hat{F}_2$ is a first difference of F_2 computed at $(i+U, j+V)$.

Upon projection, the approximate disparity (U,V) is truncated to the nearest integer. This allows for the use of a single set of first difference operators. Any reduction in accuracy is corrected by the subsequent updating process. Other techniques can be used including (1) no truncation, with interpolated first difference operators, or (2) truncation to some subpixel precision, with a finite set of first difference operators. The tradeoffs involved in the selection of such technique are discussed in [Glazer 87, Chapter 5, Section 4.4.3].

The first difference operators δ_x and δ_y specified in Section 3.2.1 include some smoothing. Smoothing is also introduced into the computation of $\hat{F}_2 - F_1$ using the filter

$$G_2 \triangleq \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Thus, the three gradients are computed using three pairs of 3×3 neighborhood operators, one of the pair about a point $f_1(i,j)$ and the other about $f_2(i+U, j+V)$.

These three gradients cannot be computed if either of the 3×3 neighborhoods does not lie completely within its respective image. This is the first error condition that may occur, called **search area overflow**. At pixels where this occurs, the gradients are set to $(0,0,0)$.

4.4 Edge Flow

The edge flow is computed as specified by Equation 16. Three error conditions may be encountered at any given pixel. First, if search area overflow has occurred then there are no gradient values available. Second, the spatial gradient may either be zero or too small to give an accurate answer when used as a divisor. The third error condition occurs when the edge flow vector is too large. The derivation of this constraint is presented in the next paragraph. For all three error conditions we set $(u_0, v_0) = (0,0)$.

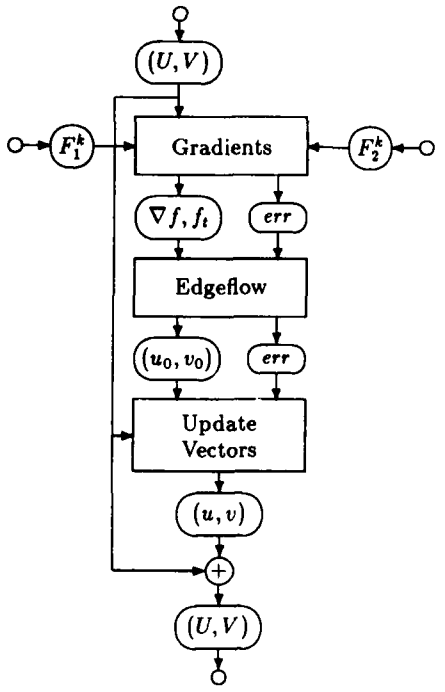


Figure 6: Hierarchical disparity: refinement

The REFINEMENT process performed at each level inputs an estimated disparity (U,V) and image pyramid data $\mathcal{F}_1^k$ and $\mathcal{F}_2^k$. Single-level (non-hierarchical) image operators compute (1) the spatial gradients and image differences $\nabla f, f_t$; (2) the edge flow at each pixel (u_0, v_0); and (3) an update vector (u,v). The update vector is added to the estimated disparity to give the updated (refined) disparity. An array of error flags (*err*), one per pixel, is carried along through the operations at one level. The operators shown may generate error flags or respond to previously generated ones.

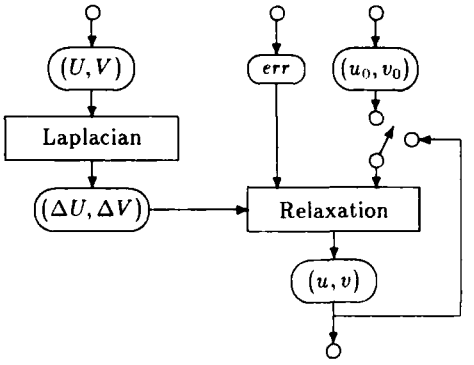


Figure 7: Hierarchical disparity: relaxation

The update vector is computed by iterative application of the relaxation operator specified in Equation 17 (when the error flag is set to NO ERROR). The edge flow (u_0, v_0) is used as the first estimate of the update vector input to RELAXATION. Thereafter, the current values of (u,v) are used. The LAPLACIAN of (U,V) is only computed once.

The third error condition is a consistency check on the computed edge flow. In Section 3.1.2 we noted that because (1) the edge flow must be smaller (in magnitude) than the update vector, and (2) availability of an approximate disparity implies a bound on the size of the update vector, for these reasons an upper bound is placed on the size of the edge flow vector. We can use either of the following relationships:

$$\|(u_0,v_0)\|_2 \leq \|(u,v)\|_2 \quad (25)$$

$$\|(u_0,v_0)\|_2 \leq \sqrt{2}\,\|(u,v)\|_{Max} \quad (26)$$

where $\|(x,y)\|_2 \triangleq (x^2+y^2)^{1/2}$ is the Euclidean norm and $\|(x,y)\|_{Max} \triangleq \max(|x|,|y|)$ is the Maximum norm. The second inequality follows from the first and from the fact that $\|(u,v)\|_2 \leq \sqrt{2}\,\|(u,v)\|_{Max}$ (since $0 \leq u^2+v^2 \leq 2[\max(|x|,|y|)]^2$). The second inequality is used when we have bounds on the individual components of the update vector. For example, the truncation of the approximate disparity vectors to the nearest pixel upon projection introduces an error of $\pm\frac{1}{2}$ pixel in each of its components. Were this the only error in that disparity estimate, then the bound on the size of the update vector would be $\|(u,v)\|_{Max} \leq \frac{1}{2}$ and using Equation 26 we get the constraint $\|(u_0,v_0)\| \leq \frac{\sqrt{2}}{2}$. Thus we have established a maximum allowable value EF_M of the magnitude of the edge flow $\|(u_0,v_0)\|_2$.

4.5 Relaxation

Update vectors are computed by an iterative relaxation process specified by Equation 17. If any of the possible error conditions exist at a pixel, then no adequate constraint line exists at that pixel. Thus the update equations cannot be used as is. However, as we noted in Section 3.3, the update equation can be interpreted as specifying that we (1) compute an update vector based on the average of neighboring information, followed by (2) projection of this vector towards the constraint line, then we see that step 1 can still be performed. Thus we arrive at an update algorithm in which at each pixel one of two things takes place: either (1) update by Equation 17 at non-error points, or (2) update by simple smoothing at points with errors. The latter is a form of interpolation into points (or areas) with no motion-based constraints. If there are large blocks where updating by (1) cannot take place, then vectors in those regions are determined from the constrained vectors which surround the respective regions.

The average update vectors $(\hat{u}, \hat{v})$ are computed using the following local averaging filter.

$$\frac{1}{12}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 0 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

The discrete Laplacian used to compute $(\Delta U, \Delta V)$ is

$$Lap_2 \triangleq \begin{bmatrix} 1/4 & 1/2 & 1/4 \\ 1/2 & -3 & 1/2 \\ 1/4 & 1/2 & 1/4 \end{bmatrix} = \frac{1}{4}\begin{bmatrix} 1 & 2 & 1 \\ 2 & -12 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

5 Experiments

5.1 The Failure of Single Level Methods

As we noted in the introduction to this chapter, gradient-based disparity computations will fail when the disparities are large relative to the range over which the image "function" is approximately linear. In general, this is a relatively short distance. Such failure takes the form of incorrect disparity estimates based on a model of the local image neighborhood that provides a bad approximation.

The following series of experiments show the inability of single level gradient-based methods to compute disparity fields. Four experiments are performed with disparities which start at a low magnitude and increase by a factor of two for each subsequent experiment. We expect the disparity computations with single gradient based methods to become more errorful as the disparities increase in size. This is in fact what the experiments show.

Two 128×128 pieces of the MANDRILL image are used with a disparity of (-5,7) from frame 1 to frame 2, i.e. 5 pixels up and 7 to the right. Low-pass pyramids were built from these images and four separate runs of single level disparity computation were performed at levels 4 (16^2) through 7 (128^2). Thus the disparities at each level were $(-.625, .875)$, $(-1.25, 1.75)$, $(-2.5, 3.5)$, and $(-5, 7)$ respectively. The EDGE FLOW TOO BIG error condition was included with maximum allowable edge flow magnitudes of 1.5, 3, 6, and 12 respectively. When this error condition is not included, results are worse.

The results are shown in Figures 8 and 9. In Figure 8, for all four levels, we show: (left) the first frame of image data; (center) disparity vectors after 50 iterations; and (right) error flags. The light gray pixels on the border of the error flags display indicate the SA OVERFLOW error condition. The predominant dark pixels throughout the interior of the error flags display indicate the EDGE_FLOW_TOO_BIG condition. In Figure 9 the statistics of the computed disparities are compared to the expected values.

It is clear that as the disparity is increased relative to the pixel spacing, the performance of single level gradient-based methods deteriorates quickly. At level 4, with expected disparity equal to $(-.625, .875)$, results are good. At level 5, with expected disparity equal to $(-1.25, 1.75)$, accuracy is significantly diminished. At level 6, with expected disparity equal to $(-2.5, 3.5)$, computed disparities are largely incorrect. At level 7, with expected disparity equal to $(-5, 7)$, computed disparities cluster near $(0,0)$ and show little relation to the expected disparity.

5.2 Hierarchical Method

The hierarchical gradient-based disparity algorithm is demonstrated in the following experiment. Again 128^2 pieces of the MANDRILL image are used with a disparity of $(-5, 7)$. A low-pass pyramid was built for each input image. Processing was begun at level 4.

In Figure 10 disparity vectors at different stages of the computation are shown. The edge flow at level 4 is shown in Figure 10a. It is labeled as iteration 0 since it is used as the initial approximation. The update vectors after 10 iterations are shown in Figure 10b. Up to this point, this experiment is equivalent to the first single level experiment of the last section. If relaxation continues at this level to the 50th iteration, the update vectors are then the same as those shown at level 4 in Figure 8.

At level 5, Figure 10c shows both the approximate disparity passed down from level 4 and the initial (iteration 0) update vector — the edge flow. Figure 10d shows the initial disparity and the final (iteration 10) update vector. At each pixel shown (every 2nd row and every 2nd column) the approximate disparity is attached by its tail to the pixel in frame 1 that it corresponds to. The tail of the update vectors are attached to the head of the approximate disparity vectors and have an arrowhead at their head. The arrowheads are proportional in size to the update vector and do not show up when that vector is small. The sum of the approximate and the update vectors is the updated disparity estimate, shown in Figure 10e.

Figures 10f–h show the comparable vectors at level 6. The final results at level 7 are shown in Figure 10i.

In Figure 11 the error flags are shown. Light gray pixels around the borders are SA OVERFLOW errors. Dark pixels are mostly EDGE_FLOW_TOO_BIG errors. There were 2, 8, 188, and 1021 EDGE_FLOW_TOO_BIG errors at levels 4, 5, 6 and 7 respectively and 1 ZERO_GRAD error at level 7.

In Figure 12, histograms of the disparity fields at each of the levels are shown. These histograms are built by counting all disparity vectors that lie within $\pm\frac{1}{2}$ of each pixel. For example, the number at location (0,0) in each histogram is the number of disparity vectors with row and column components both in the range $[-\frac{1}{2}, \frac{1}{2})$.

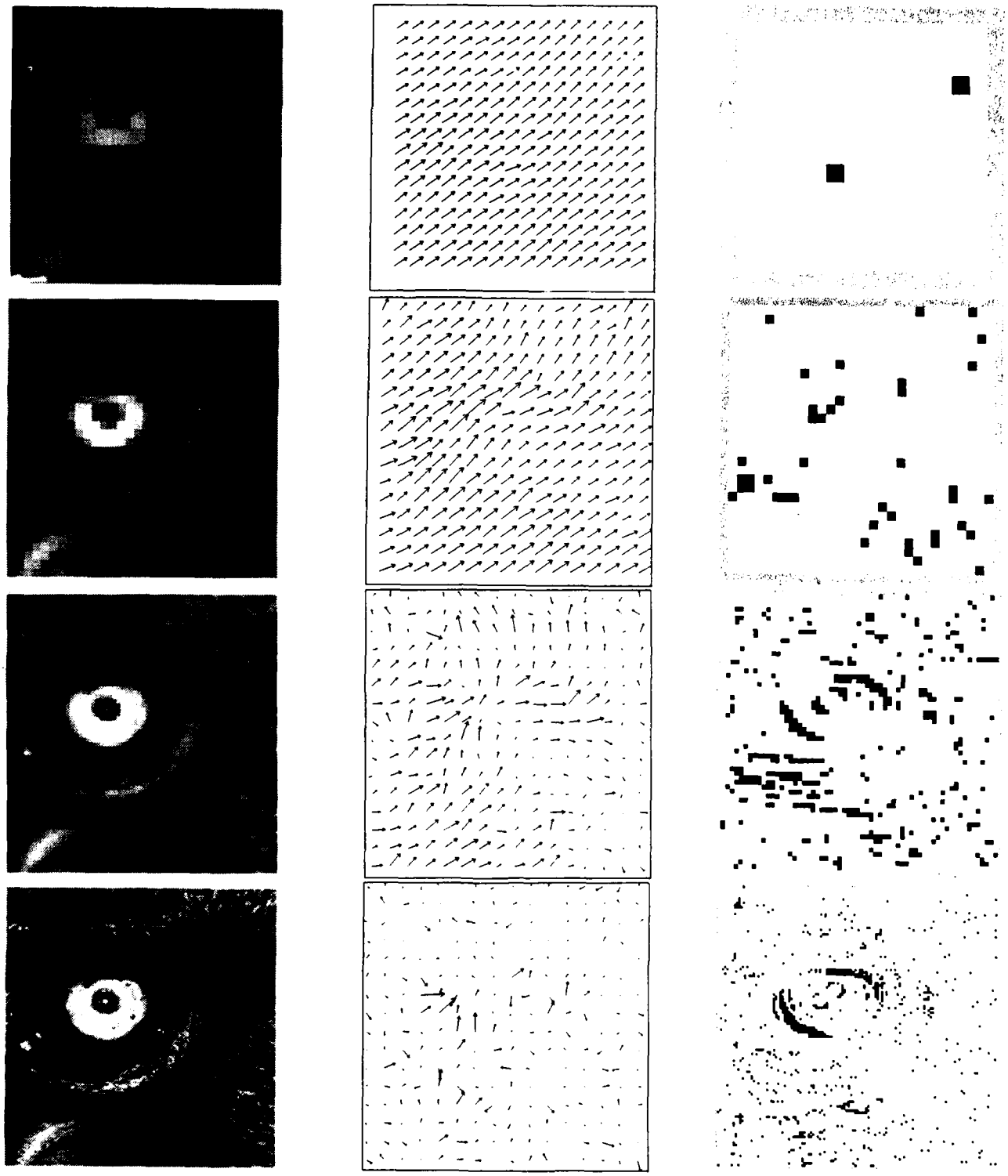


Figure 8:
Single-level analysis

1st row : level 4, 16 × 16
2nd row : level 5, 32 × 32
3rd row : level 6, 64 × 64
4th row : level 7, 128 × 128

1st column : the first frame of image data
2nd column : disparity vectors after 50 iterations
(only a 16 × 16 subset is shown)
3rd column : error flags

Level 4

Actual	D_{row}	D_{col}
mean	−.579	.820
SD	.066	.095
minimum	−.749	.585
maximum	−.364	1.037
Expected	−.625	.875

Level 5

Actual	D_{row}	D_{col}
mean	−1.112	1.464
SD	.284	.407
minimum	−2.321	.471
maximum	−0.317	3.184
Expected	−1.25	1.75

Level 6

Actual	D_{row}	D_{col}
mean	−0.924	1.01
SD	1.066	1.26
minimum	−5.362	−3.92
maximum	4.538	5.19
Expected	−2.5	3.5

Level 7

Actual	D_{row}	D_{col}
mean	−0.47	0.50
SD	1.74	1.69
minimum	−11.03	−7.42
maximum	−7.28	10.55
Expected	−5.	7.

Figure 9: Single-level analysis: Disparity Statistics

In Figure 12, disparity vectors are included in the histograms only if no error has been detected at their respective pixel locations. The expected disparities are $(-.625, .875)$, $(-1.25, 1.75)$, $(-2.5, 3.5)$, and $(-5, 7)$ at levels 4 through 7 respectively. The center of gravity of these histograms is one measure of accuracy. For the histograms shown, the centers of gravity are $(-.8918, .5309)$, $(-1.782, 1.384)$, $(-2.919, 3.401)$, and $(-4.929, 6.659)$ at levels 4 through 7 respectively. In all cases, both components are within $\pm\frac{1}{2}$ pixel spacing of the expected disparity values.

In Figure 13, the statistics of the row and column components of the disparity vectors are shown. Again only vectors at non-error pixels are included. The mean values are not equal to the above listed centers of gravity of the histograms, because the histograms represent disparity vectors with components truncated to the nearest integer. These can be compared to the corresponding statistics for the single level shown in Figure 9. The hierarchical method clearly succeeds where the single level method fails.

6 Summary and Discussion

In this report we have demonstrated that *first order gradient-based methods may be extended to the computation of large disparities by formulating a hierarchical generalization of the single level method that operates on low-pass image pyramids.* A thorough formal analysis of the gradient-based updating equations was presented as well as the implementation of the method in a hierarchical processing cone algorithm. Experiments were presented both to show how single level methods fail and how the hierarchical method succeeds.

The hierarchical gradient-based algorithm uses a coarse-to-fine control strategy to progressively refine individual motion estimates. This strategy can also be applied to correlation-based motion detection algorithms [Wong & Hall 78, Moravec 81, Glazer *et al.* 83, Anandan 84, Quam 84]. Two specific hierarchical algorithms, one correlation-based [Glazer 87, Chapter 4] and the gradient-based algorithm presented here, have been shown by experiment to have comparable accuracy [Glazer 87]. Furthermore, comparison of the computational costs, both arithmetic and data transfer, show that the gradient-based algorithms are, in general, less costly [Glazer 87, Chapter 7]. The computational costs of hierarchical correlation are relatively high for two reasons: (1) many multiplications and additions are needed for the correlation inner product; and (2) many pixel values must be transferred over multi-pixel distances in the image plane.

References

[Anandan 84] Anandan, P., Computing Dense Displacement Fields with Confidence Measures in Scenes Containing Occlusion, *SPIE Vol. 521 Intelligent Robots and Computer Vision*, Cambridge, Mass., 1984.

[Burt 81] Burt, P.J., Fast Filter Transforms for Image Processing, *CGIP* **16**:20–51, 1981.

[Burt 82] Burt, P.J., Pyramid-Based Extraction of Local Image Features with Applications to Motion and Texture Analysis, *Proc. SPIE Conf. on Robotics and Industrial Inspection*, San Diego, 1982.

[Cornelius & Kanade 83] Cornelius, N. and Kanade, T., Adapting Optical-Flow to Measure Object Motion in Reflecting and X-ray Image Sequences, *Proc. ACM SIGGRAPH/SIGGART Interdisciplinary Workshop on Motion: Representation and Perception*, pp. 50-58, 1983.

[Courant & Hilbert 53] Courant, R. and Hilbert, D., *Methods of Mathematical Physics, Volume I*, Interscience Publishers, Inc., New York, 1953.

[Fennema & Thompson 79] Fennema, C.L. and Thompson, W.B., Velocity Determination in Scenes Containing Several Moving Objects, *CGIP* **9**(4):301–315, 1979.

[Glazer 81] Glazer, F., Computing Optic Flow, *Proc. 7th. IJCAI*, pp. 644–647, Vancouver, BC, 1981.

[Glazer *et al.* 83] Glazer, F., Reynolds, G., and P. Anandan, Scene Matching by Hierarchical Correlation, *Proc. CVPR* pp. 432–441, 1983.

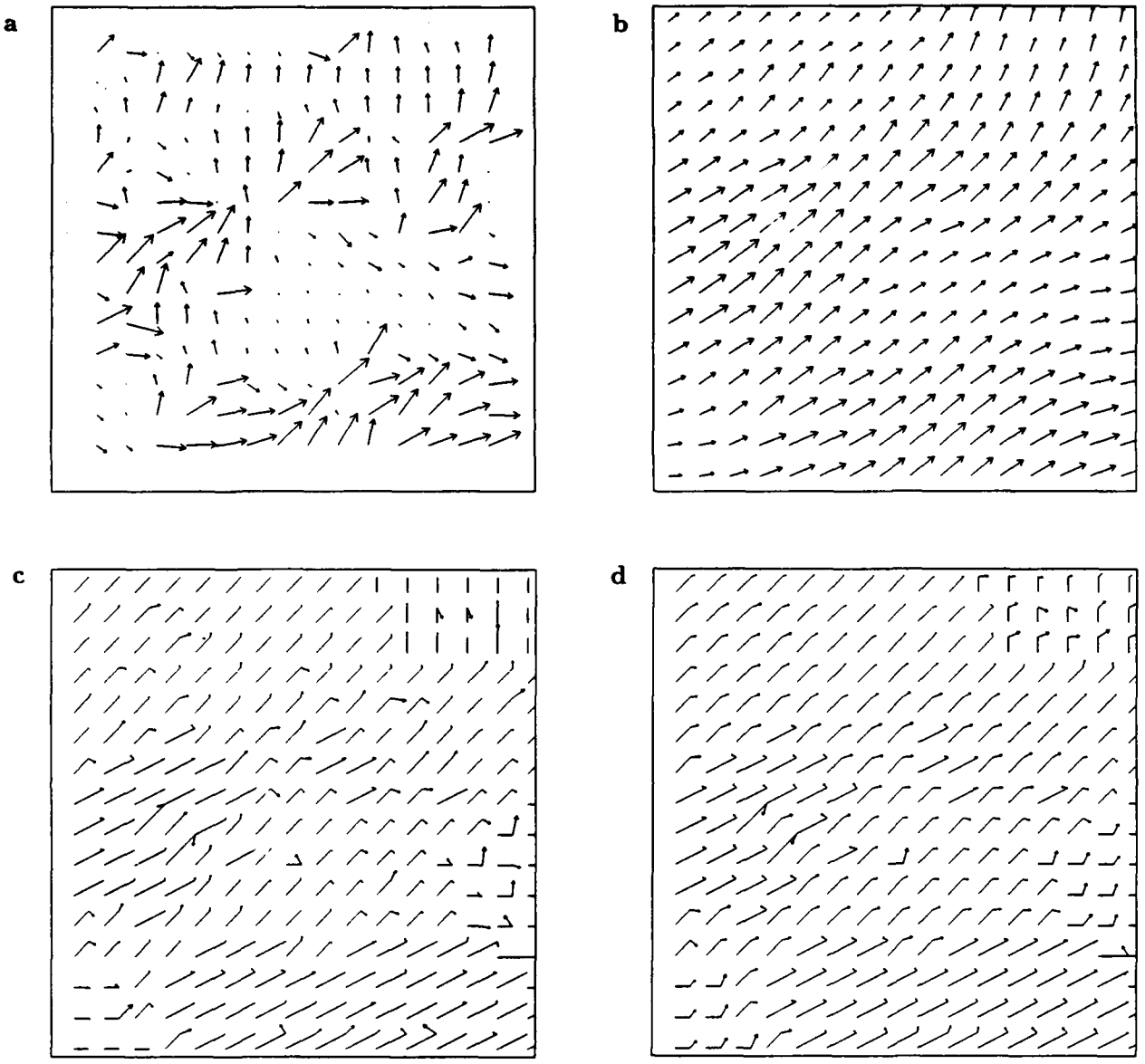
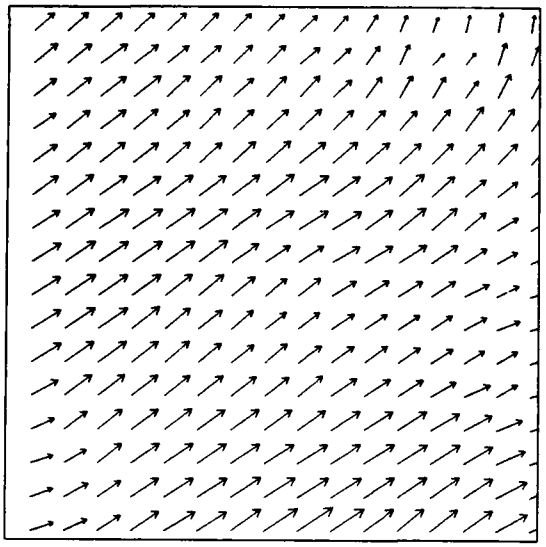


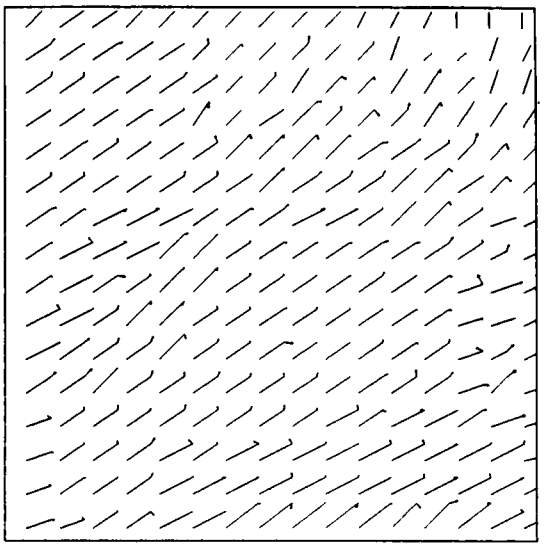
Figure 10: Multilevel experiment: disparity vectors

(a) level 4, iteration 0 (b) level 4, iteration 10 (c) level 5, iteration 0
(d) level 5, iteration 10 (e) level 5, updated vectors (f) level 6, iteration 0
(g) level 6, iteration 10 (h) level 6, updated vectors (i) level 7, updated vectors

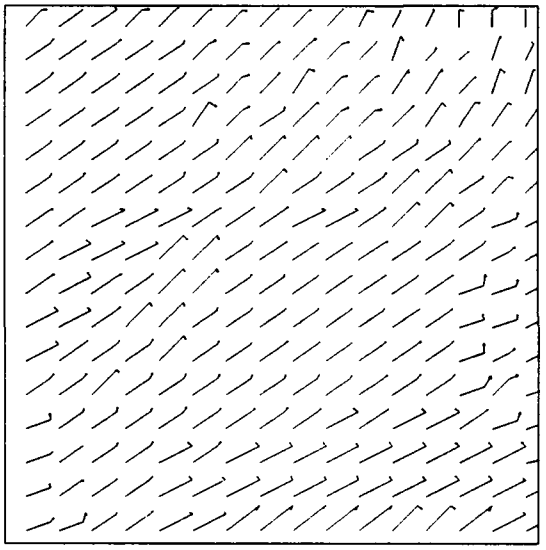
e



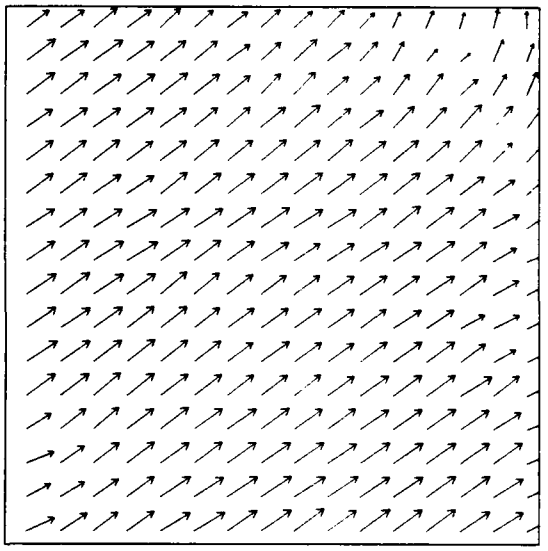
f



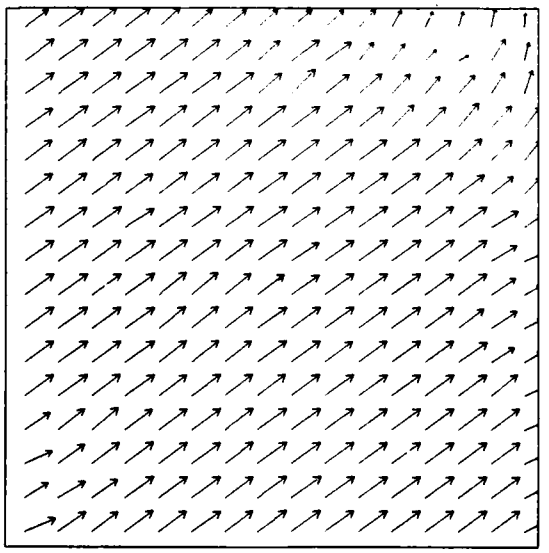
g



h



i



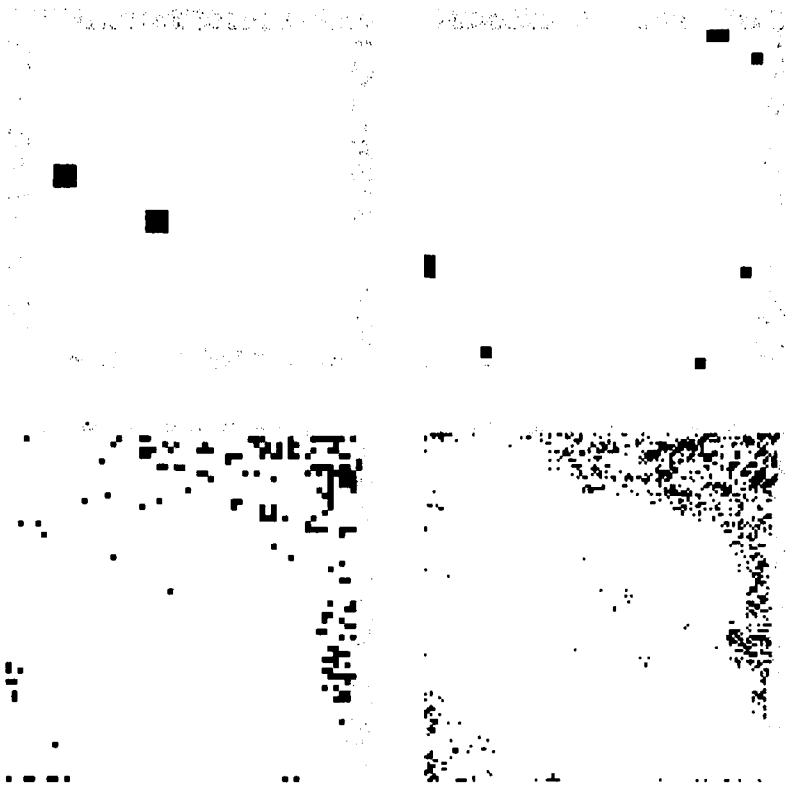


Figure 11: Multilevel experiment: error flags

Error flags at levels 4 through 7 for the multilevel experiment. Light gray pixels around the borders are SEARCH_AREA_OVERFLOW errors. Dark pixels are predominantly EDGE_FLOW_TOO_BIG errors.

[Glazer 87] Glazer, F., Hierarchical Motion Detection, Ph.D. Thesis and COINS Tech. Report 87-02, U.Massachusetts, Amherst, 1987.

[Hanson & Riseman 74] Hanson, A. and Riseman, E.M., Preprocessing Cones: A Computational Structure for Scene Analysis, COINS Tech. Report 74C-7, U.Massachusetts, Amherst, 1974.

[Hanson & Riseman 80] Hanson, A. and Riseman, E.M., Processing Cones: A Computational Structure for Image Analysis, In: *Structured Computer Vision*, Tanimoto, S. and Klinger, A. (Eds.), Academic Press, New York, 1980.

[Horn & Schunck 81] Horn, B.K.P. and Schunck, B.G., Determining Optical Flow, *Artificial Intelligence* **17**(1-3):185-204, 1981. Also in *Proc. IUW (DARPA)*, April, 1981.

[Kahn 85] Kahn, P., Local Determination of a Moving Contrast Edge, *IEEE Trans. PAMI* **7**(2):402-409.

Moravec 81] Moravec, H.P., *Robot Rover Visual Navigation*, UMI Research Press, Ann Arbor, Michigan, 1981.

Nagel 83] Nagel, H.-H., Displacement Vectors Derived from Second Order Intensity Variations in Image Sequences, *CVGIP* **21**:85-117, 1983.

Quam 84] Quam, L.H., Hierarchical Warp Stereo, *Proc. IUW (DARPA)*, pp. 149-155, October 1984.

Schunck 83] Schunck, B.G., Motion Segmentation and Estimation, Ph.D. Thesis, MIT, Dept. of EE & CS, 1983.

Tanimoto & Pavlidis 75] Tanimoto, S., and Pavlidis, T., A Hierarchical Data Structure for Picture Processing, *CGIP* **4**(2):104-119, 1975.

Wong & Hall 78] Wong, R.Y. and Hall, E.L., Sequential Hierarchical Scene Matching, *IEEE Trans. Comp.* **27**(4):359-366, 1978.

Yachida 81] Yachida, M., Determining Velocity Map by 3-D Iterative Estimates, *Proc. 7th. IJCAI*, pp. 716-718, Vancouver, B.C., Canada, 1981.

Level 4

	0	1
-1	81	92
0	10	11

Level 5

	0	1	2
-2	9	355	326
-1	2	166	24
0	0	0	0

Level 6

	0	1	2	3	4
-4	0	2	4	2	0
-3	1	19	66	1575	1561
-2	1	21	18	197	35
-1	0	4	3	3	1
0	0	0	0	0	0

Level 7

	0	1	2	3	4	5	6	7	8
-8	0	0	1	2	0	0	0	0	0
-7	0	1	4	2	4	3	0	0	0
-6	0	8	19	15	27	74	62	7	2
-5	2	5	27	60	90	248	1263	11195	4
-4	3	24	37	25	65	118	206	95	6
-3	3	15	29	19	10	36	88	40	1
-2	0	1	13	5	8	17	8	5	0
-1	0	0	3	0	1	1	0	0	0
0	0	0	0	0	0	0	0	0	0

Figure 12: Hierarchical method : disparity histograms

These are two-dimensional histograms of the disparity fields at levels 4 through 7 for the multilevel experiment. At each level, the pixel location nearest the expected disparity level is surrounded by a box. (At level 6 four pixels are equidistant from the expected disparity, so the box surrounds all four.)

Level 4

Actual	D_{row}	D_{col}
mean	−.488	.628
SD	.144	.189
minimum	−.794	.161
maximum	−.107	.980
Expected	−.625	.875

Level 5

Actual	D_{row}	D_{col}
mean	−1.112	1.489
SD	.180	.285
minimum	−1.510	.284
maximum	−0.230	2.202
Expected	−1.25	1.75

Level 6

Actual	D_{row}	D_{col}
mean	−2.343	3.195
SD	.255	.447
minimum	−3.552	−0.191
maximum	−0.385	4.007
Expected	−2.5	3.5

Level 7

Actual	D_{row}	D_{col}
mean	−4.872	6.614
SD	.421	.868
minimum	−7.862	.081
maximum	−0.917	7.852
Expected	−5.	7.

Figure 13: Hierarchical method: disparity statistics

These statistics are computed over those pixels at which there were no error conditions. This included 194, 882, 3513, amd 14007 pixels at levels 4 through 7 respectively (out of a possible 256, 1024, 4096, and 16384). These are the unmarked pixels in Figure 11

SHAPE RECONSTRUCTION ON A VARYING MESH

Isaac Weiss
Center for Automation Research
University of Maryland, College Park, MD 20742

ABSTRACT

A central class of IU problems is concerned with reconstructing a shape form an incomplete data set, such as fitting a surface to (partially) given contours. We present a new theory for solving such problems. Unlike the current heuristic methods, we start from fundamental principles that should be followed by any reconstruction method, regardless of its mathematical or physical implementation. We then present a mathematical procedure which conforms to these principles. One major advantage of the method is the ability to handle shapes containing variations of different scales. A sharp variation, such as a corner, requires a high resolution mesh for adequate representation, while slowly varying section can be represented with sparser mesh points. Unlike current methods, our procedure fits the surface on a varying mesh. The mesh is constructed automatically to be denser at parts of the image having more rapid variation. Analytical examples are given in simple cases, followed by numerical experiments.

I. INTRODUCTION

A central problem of Image Understanding is the inference of a 3-D shape from given 2-D images. According to the types of information that can be extracted from the images, theories have developed which can be classified under categories such as Shape from Stereo, Shape from Motion, Shape from Contour, Shape from Shading, Shape from from Texture, etc. A reliable shape reconstruction mechanism will, apparently, have to combine several of these kinds of clues in a consistent way. The present paper addresses mainly the Shape from Contour component of the problem. However, our theory is based on a general extremum principle which can be used to incorporate most of the other components of surface reconstruction as well. Related topics, such as smoothing and interpolation, are also treated in this framework.

The Shape from Contour problem can be stated as follows: Given a set of contours, such as may be provided by some image processing device, one wants to infer the shape of the surface that is most likely described by these contours. One can also consider an inverse, or a complementary problem: given the surface, one seeks its "natural parametrization", namely contours that will convey its essential characteristics in the most economical, yet reliable way. One known mechanism that performs these tasks well is the human visual system. A few drawn lines can create a surprisingly vivid and convincing impression of a 3-D shape [Marr, 1982; Barrow & Tenenbaum, 1981].

In this paper, we address both sides of this problem, and thus we shall refer to it as "Shape Representation by Contours", which includes shape from contour as well as contour from shape. The treatment is on the fundamental and general level of first, finding the principles that must guide any process of representing a surface by contours, regardless of its physical implementation, and second, proposing a mathematical mechanism that can perform these tasks, conforming with our principles. We assert that any good visual system, when viewed as a "black box", should behave in a way similar to our mathematical procedure. We do not attempt to specifically emulate the human visual system. Nevertheless, as the eye is the most successful image processing system we can use, we shall test the performance of our abstract procedure against its performance. In the following we shall briefly review some of the prior work and then summarize the requirements that we impose on a surface reconstruction mechanism.

Much of the previous work has been concerned with various subsets of the general problem. One such subset is the interpolation of a 2-D curve, given points in a 2-D plane. The points are assumed to lie in the vicinity of the curve, but not necessarily on it. The common approach the problem is through an extremum principle. In the simplest case, one assumes that the curve is a straight line, and performs a least square fitting. For general curves, one possibility [Horn, 1977] is to assume that the interpolating curve has the minimum overall curvature, subject to the given data constraints. For instance, given the coordinates of two end-points of a curve and its tangents there, one can find the curve $\vec{t}(s)$, where $\vec{t}$ is the tangent and s is the arclength, that minimizes the functional $E = \int (d\vec{t}/ds)^2$, over the domain of functions that satisfy the given boundary conditions. (This derivative of the tangent is the curvature $\vec{k}$.) By a physics analogy, this is also known as the minimum energy principle. If the curve of least energy is nearly straight, it can be approximated by cubic spline polynomials spanned between some set of mesh points.

Smoothing is a problem closely related to interpolation and reconstruction. We can regard the noisy curve (or surface) as initial data from which a smooth curve is to be constructed. Thus a good reconstruction technique can also be used as a smoothing method, and as such it may have a very wide applicability in computer vision, beyond shape reconstruction. The converse is not true, however. A reconstruction method must be able to deal with sparse data input, e.g. to bridge gaps between known curves or points, or build surfaces from sparse contours. These contours do not need to be smooth and may contain corners or bumps. In short, we regard interpolation and smoothing as overlapping subsets of the reconstruction problem.

A problem shared by all known smoothing techniques, and by the above minimum curvature principle in any of its applications, is the handling of sharp variations in the curve such as corners or bumps. A technique that efficiently removes small perturbations like noise also blunts corners which may be an essential part of the shape. A general framework for smoothing a 3-D surface is given in [Rosenfeld and Kak, 1982]. It consists of minimizing a cost function ϕ, which can be a weighted integral over the picture of $(\frac{\partial f}{\partial x})^2 + (\frac{\partial f}{\partial y})^2 + (f - g)^2$, where f is a possible smoothing of a noisy picture g. Intuitively, this generalizes the idea that Grimson [1981] called informally "no news is good news", namely that the overall change in the surface, i.e. its derivatives, or some curvature, is minimal unless we have good evidence to the contrary. Of course, the exact form of ϕ has to be given, and it involves a trade-off between noise removal and corner preservation.

The smoothing technique derived from our general reconstruction formalism is a departure from the above formalism. By reparametrization of surface we will able to alleviate or eliminate this need for a trade-off of noise vs. blunted corners and deal with curves with sharp turns.

Some *ad hoc* variation of the minimum curvature principle have been applied to 3-D problems. Barrow and Tenenbaum add torsion to the functional E containing the curvature, but the torsion, being a higher derivative, aggravates the sensitivity to noise. Terzopoulos [1982], [1985] regards a surface as a thin flexible plate under tension, attached to data points by "springs". He minimizes the energy of such a system, which is a functional of the surface and its derivatives. This amounts to choosing a simple form of ϕ, and adding first derivatives of f, with parameters reflecting the above trade-off. These parameters need to be adjusted empirically for each application, and the restriction to a thin plate (one that is not too different from a plane) limits the model's usefulness.

Another subset of the 3-D reconstruction problem to which the minimum curvature principle has been applied to is (planar) "deprojection": given a curve on a plane which is slanted and tilted relative to the the observer's line of view, estimate the most likely slant and tilt. (The observer can only see a projected image of the curve.) Humans, for example, when asked to interpret an apparent ellipse as a slanted shape, will regard it as a slanted circle. On this assumption the slant and tilt can be calculated. According to the minimum curvature principle, the slant and tilt are those which minimize the overall curvature of the "deprojected" curve.

Several problems arise in these applications. First, a very high penalty is incurred by corners and bumps on the curve. Since the curvature of a sharp corner tends to infinity, a corner will tend to dominate the curvature integral. Straight lines, having zero curvature, will be ignored, regardless of their length. Thus the minimum curvature principle will be unable to handle curves with corners or even small, sharp bumps, that may be caused by noise, and long straight lines. Second, the curvature measure is not dimensionless. The interpolated, or deprojected curve will depend on the size of the image. Third, in the case of the slant finding problem, the deprojected curve is not in agreement with human perception. The deprojection of an ellipse will not be a circle.

A mechanism which partly solves the problems of the minimum curvature principle is the Brady-Yuille [1984] maximum compactness measure. Applied to the slant finding problem of a closed planar curve, it states that the deprojected curve is the one for which the ratio of the area enclosed by the curve to its perimeter squared is maximal. It is related to the statistically-based Maximum Likelihood Method of Davis, Janos and Dunn [1982] and Witkin [1981]. These measures are inherently insensitive to noise, are dimensionless and gives the expected intuitive results. These authors did obtain the circle as the deprojected ellipse, and they also calculated the resulting slant and tilt. It seems hard, however, to apply these measures to more general curves.

The experience with these investigations over the past few years enables us to formulate some requirements that a surface reconstruction mechanism should satisfy, and it shows the need for a new mechanism that will meet them. To meet these demands, our proposed formalism had to depart from the conventional form of ϕ above, by redefining length and curvature and by introducing a reparametrization of curves and surfaces. We now list these requirements and touch on the way our formalism deals with them.

1) Surface consistency: A surface reconstruction mechanism will build a surface that does not change very much, unless this mechanism has information indicating such a change, such as boundaries or other contours. should not add extraneous information of its own. We shall define a functional of the surface, related to the curvature, that the desired surface should minimize.

2) Translational and rotational invariance—i.e. if the input image is rotated or translated, the output should move by the same amount but not change its shape.

3) Dimensionlessness, or scale invariance: When the distance between an object and a viewer changes, the object's apparent size changes, but not its shape. Therefore, a reconstruction mechanism should yield the same output shape from an input image, regardless of its size. Thus the mathematical method of reconstruction must be dimension-

less, or invariant to scale. Partly to solve this problem, we have introduced a renormalization of the arclength.

4) Handling of different variation scales: As mentioned above, the mechanism has to handle both rapid curvature changes, such as sharp corners, without blurring them too much, while also taking into account stretches of slow variation, such as straight or slightly curved parts of the shape. All the schemes based on simple derivatives of the surface have severe difficulties in this respect. A reparametrization of the curves, derived from the extremum principle itself, helps us to solve this problem.

5) The mechanism should have a wide domain of applicability, including closed and open space curves and surfaces, with few or no restrictions.

The following requirements do not seem to be essential but have various degrees of desirability:

6) Computational efficiency: This is, of course, a function not only of the mathematical scheme, but also of the specific implementation technique and the hardware architecture, which are issues by themselves. We can generally say that since the bulk of the computation in our theory is done on local variables, the scheme lends itself to implementation on parallel computers. As the general theory is non-linear, slowing down is expected compared to the special cases in which it can be linearized, such as "thin plates".

7) One would prefer the same mechanism to handle both aspects of the problem mentioned above, namely finding both the surface and a suitable set of coordinates on it. While previously suggested mechanisms only attempt to find the surface, ours also lead to a "natural" parametrization of it.

8) Consistency with human perception: For special cases we can obtain analytic results and compare them with human perception, with good agreement. For example, a circle which is known to be an occluding boundary will give rise to a sphere.

The first part of the paper describes the theory. We start with curves in the 2-D plane, and continue with general surfaces in 3-D. The second part describes numerical experiments.

II. SMOOTH CONTOURS

In this section we shall deal with the particular case in which a simple minimum curvature principle gives at least mathematically definite and stable results, namely smooth curves, without noise or corners. We shall modify it in a way that will make it dimensionless, and will also make it amenable to treatment by a standard analytic minimization technique. We shall use this technique to find the equation for the curve of minimal modified curvature that passes between two end-points. We shall also obtain, as a byproduct, the circle as the "deprojection" of the ellipse. We will preserve the useful properties of the energy principle.

As curvature is a good measure of surface change, the mechanism will seek a surface which has the minimal overall (modified) curvature, while being consistent with the available information. It is also qualitatively clear that information is closely related to the smoothness of the curve or the surface. A straight line can be described by the coordinates of the end-points only, while a more complicated shape will require more information. Therefore, minimizing curvature is in accordance with minimizing the extraneous information generated by the mechanism itself.

As one wants to represent the amount of information contained in a curve by its curvature or a related quantity, the natural mathematical quantities to deal with are the derivatives of the tangent vector, such as the curvature vector $\vec{k} = d\vec{t}/ds$. As we demand rotational invariance, we have to use a scalar product of these vectors. This leads to the suggestion of the principle of minimum "spline bending energy" [Horn, 1981], by which one wants to minimize the integral

$$E_{spline} = \int \vec{k}^2 ds = \int \left(\frac{d\vec{t}}{ds}\right)^2 ds$$

taken over the whole curve. This expression approximates the bending energy of a flexible stick, such as a "spline" used by draftsmen to draw a curve between points.

This measure has been implemented in several ways. First, in an interpolation problem: given the coordinates and tangents of two points of the curve, one can find the curve that will pass through them and will minimize this integral. Second, in a deprojection problem: given a closed boundary such as an ellipse, one can try to interpret it as a projected image of a surface in 3-D. The human eye will usually regard an ellipse as a slanted circle, and we would like an extremum principle to yield the circle as an extremum over the set of all curves compatible with the projected ellipse. This would be consistent with the result obtained by the Maximum Likelihood method. Limiting ourselves to planar curves (zero torsion), this compatible set is the set of ellipses (produced by different slant angles), with the same major axis as the apparent one.

The minimum curvature principle fails in this task as it does not lead to the circle as the extremum over the set of ellipses. Even worse, as it is not dimensionless, the extremum will depend on the apparent size of the image, contrary to to our demands.

We now propose a modification of the minimum curvature principle, which will work for smooth curves. Letting L be the total length of the curve, we can define the dimensionless variable

$$l = \frac{s}{L}$$

where s is the length of the curve lying between a point on the curve and one of its ends. l is a measure of the relative position of a point on the curve, regardless of the curve's length. We define the "normalized" energy E_n as the integral

$$E_n = \int_0^1 \frac{d\vec{t}}{dl} \cdot \frac{d\vec{t}}{dl} dl$$

The difference between E_{spline} and the normalized energy E_n is the replacement of ds by dl. This is equivalent to multiplying the "spline energy" by L. As both l and $\vec{t}$ are dimensionless, so is E_n.

Now, in accordance with our modified minimal curvature requirement, we shall look for a curve, $\vec{t}(l)$, that will extremise the normalized energy E_n over the set of all possible curves having the same $\vec{t}$ at the end-points.

An important feature of the normalized arclength l is that its values at the end-points are fixed, in this case to 0 and 1. This is in contrast to the original arclength s, whose value at one end is the length of the curve, and it is more in the character of an ordinary coordinate such as x or y. This makes it possible to use l as the independent variable in many mathematical techniques for which s in not suited. In particular, we are interested now in techniques from the calculus of variations for the minimization of the energy integral. Other works have also used these techniques, but only after assuming that the curve (or surface) is nearly straight, so that s could be approximated by x.

We will deal in this paper with functionals E which depend on several functions $\psi_i(x^k)$, such as curve or surface tangents, and their derivatives. Here $x^k = x^1 \dots x^n$, where n is the number of independent variables. A necessary condition for ψ_i to minimize the functional E (over some domain of admissible functions) is that for all infinitesimal variations $\delta\psi_i$, the infinitesimal variation of the functional δE vanishes. Namely, for the given function F of ψ_i and their derivatives $\partial\psi_i/\partial x^k$, and and for all $\delta\psi_i$ one must have

$$\delta E = \delta \int F(\psi_i, \frac{\partial \psi_i}{\partial x^k}) dx^n = 0$$

where dx^n is the volume element in the n-D space.

If we restrict ourselves to the domain of (sufficiently differentiable) functions with fixed boundaries, such as curves with known coordinates and tangents at their ends, then the above equation will be satisfied if, for each i, F satisfies the Euler-Lagrange equation [Courant and Hilbert, 1953]:

$$\frac{\partial F}{\partial \psi_i} - \sum_{k=1}^{n} \frac{\partial}{\partial x^k} \left(\frac{\partial F}{\partial (\partial \psi_i / \partial x^k)} \right) = 0$$

In our normalized energy functional E_n, the function F is a simple quadratic, the independent variable is l, and the unknown function variables ψ_i are the components of $\vec{t}$, t_x and t_y. These unknowns are not independent, though. As $\vec{t} = d\vec{x}/ds$ we have $t_x^2 + t_y^2 = 1$. By using a polar coordinate system, with the angle ϕ defined as the angle between the tangent $\vec{t}$ and (say) the x axis, we will obtain an equation of a single function variable $\phi(l)$. In this system we have

$$t_x = \cos\phi$$
$$t_y = \sin\phi$$

The energy now reads

$$E_n = \int_0^1 \left(\frac{d\phi}{dl} \right)^2 dl$$

In the case that the only information we have about the end-points is the inclination (tangent angle) of the curve there, ϕ_1 and ϕ_2, this last expression suffices. To minimise E_n, we apply the Euler-Lagrange equation to it and obtain

$$\frac{d^2\phi}{dl^2} = 0$$

with the unique solution

$$\phi = (\phi_2 - \phi_1)\, l + \phi_1$$

which is a circular arc. Thus, the curve which minimizes E_n over the domain of functions having given tangent angles at the ends, is a circular arc.

In particular, when $\phi_1 = \phi_2 = \pi/2$, we obtain two semi-circles (upper and lower). The problem of deprojecting an ellipse is now solved trivially. Since the circle minimizes E_n over all functions with end-points perpendicular to the x axis, it will also do so over the sub-domain of ellipses with an axis along x. This last particular result can be obtained by much more elementary means, because one only needs to find the value of one parameter (the ellipse's eccentricity, related to the slant). The general interpolation problem, however, requires the full power of the calculus of variations as we are looking for a *function* that connects the two points.

In most applications one has more input data than the end tangents. One can include the spatial coordinates of the end-points as constraints, and show that we still obtain a circular arc [Weiss 1986a]. Other input information can be added as an attraction potential, as described below.

III. LARGE- AND SMALL-SCALE VARIATIONS

The simple minimum curvature principle, our modification notwithstanding, suffers from a severe drawback: It cannot handle sharp turns in the curve. A sharp corner will completely dominate the curve, and the value of the integral will depend on the exact shape of the corner, with point-like edges leading to infinities. Even a small bump will have a considerable influence, and in fact, the smaller the bump, the greater its effect on the integral will be (keeping its shape similar). This makes E_n inapplicable for curves with edges, or noise, without elaborate filtering techniques. On the other hand, straight lines are ignored by this measure, regardless of how long they may be.

We propose a new kind of an extremum principle, one of whose advantages is essentially solving this small (and large) scale variation problem. (Another advantage will become apparent when we go over to 3-D.) Our theory is based on minimizing a functional of the curve, which we call "spring energy", as it is analogous in many respects to the energy of a physical spring. In addition to the ability to bend, as in a conventional spline model, the spring has another degree of freedom. It can stretch or shrink in some uneven manner, creating a varying density of coils along it. The curve can now by reparametrized by a variable α, which represents the density of coils along the spring. In a discrete representation of the curve, this corresponds to the density of the knot points.

In the following we shall (usually) adhere to the following terminology: Knots, or knot points are a finite number of points on the curve itself, which represent it. All other

curve points can be calculated from these. Data points are given by a sensor of some kind. Grid points are a fixed network of points in n-D space, independent of the curve or the data. "Mesh points" is a general term used in place of either knots or grid points.

The spring's energy consists of three parts: (a) Bending energy, which is related to the curvature of the curve. A rapidly varying curve bends more then a slowly varying one, and thus has a higher bending energy. This part is similar to the (normaled) spline energy discussed before. (b) Stretching energy. This component represents the additional degree of freedom that a spring has. When one stretches a spring by holding it at both ends, the spring will stretch uniformly, as this is the state of minimum energy under these conditions. Any deviation from uniformity will increase this stretching energy. It is this deviation from uniformity that we are interested in, because it can be related to a nonuniform placement of the knot points. We can associate the density of the knot points with the density of the spring coils, so that a high concentration of coils in one section means a larger number of knot points there. (c) Energy resulting from external forces operating on the spring, forcing it to deviate from its natural form. We will assume that each data point is a source of an attraction force operating on each point of the spring representing the curve, with the force being stronger on curve points that get closer to the data point. In minimizing the overall energy, there is a trade-off between its three components, and it turns out that a local trade-off between the bending and stretching energies creates a desirable placement of the knots. This will be shown analytically in simple cases and will be demonstrated numerically.

Intuitively, one can regard the attraction energy (c) as the energy that forces the spring to follow the data points. There will be bending of the spring in places where the data points indicate corners. Because the bending tends to distribute itself uniformly over all coils, the coils will tend to concentrate in the regions of high bending. That makes the amount of bending *per coil* more uniform, which reduces the total bending energy (a). The stretching energy (b), on the other hand, will increase when the coils' spatial distribution deviates from a uniform one, and thus a balance is created between the tendency of the coils, or the knot points, to concentrate in high bending sections, and their tendency to be uniformly distributed. This reflects a balance between the wish to follow the curve as closely as possible, and the wish to smooth the data. Unlike other schemes, however, this balancing is done it locally at each part of the curve, and is thus much more responsive to the local shape of the curve, than in schemes which try to trade off between smoothing and accurate fitting in some global way. This knot placement method is a natural part of the minimization principle and the placement is done automatically in the course of numerically minimising the energy functional.

We turn now to the mathematical representation of the model. Denoting by E the total energy of the curve ("spring"), by $\vec{t}$ the tangent vector, by s the arclength, and by α an independent parameter along the curve, analogous to the distribution of coils along the spring, the energy is

$$E = \int_{\alpha_0}^{\alpha_1} \left[\left(\frac{d\vec{t}}{d\alpha} \right)^2 + \frac{\kappa}{L^2} \left(\frac{ds}{d\alpha} \right)^2 + \frac{V}{L} \frac{ds}{d\alpha} \right] d\alpha$$

where α_0, α_1 are the values of α at the curve's ends, κ is a constant factor related to the spring's flexibility, $L = \frac{s_1 - s_0}{\alpha_1 - \alpha_0}$ (the total arclength divided by the total change in α) is used as a normalization factor, and $V = V(\vec{x})$ is the local value of the external potential energy (described below).

The first term above is the bending energy, the second term is the stretching energy and the third term represents the influence of the external forces acting on the spring. By definition, the force acting on a particle is the gradient of (the negative of) the potential at that point. The best fitting curve is now the one that minimizes E, over a suitable space of admissible curves, subject to some boundary conditions at the end-points. Given the potential $V(\vec{x})$ and the boundary conditions, one can find the functions $\vec{t}(\vec{x}), s(\vec{x}), \alpha(\vec{x})$ which minimize the functional E, namely find both the curve and the parameter α along it.

To gain more intuitive understanding of α, we will ignore the third term for the moment, and assume that the curve $\phi(l)$ (or equivalently $\vec{t}(l)$) is given. The only unknown is its reparametrization α. It can be seen that the minimization creates two forces that influence the, reparametrization, resulting from the two first terms in E. The stretching, represented by the second term, pushes α to follow l as closely as possible (thus making the first term close to the notion of curvature). The bending, on the other hand, drives α to follow ϕ, so that a large change in ϕ, as in a corner, will cause a correspondingly large change of α in that region.

The spring is "ideal" in the sense that it, or part of it, can be shrunk to to an infinitesimal length (the coils have no thickness). This is what will happen in a corner, as we will see more precisely below. In this case, a finite bending $\Delta\phi$ is concentrated in an infinitesimally small length of the curve. To minimize the energy, a *finite* "number of coils" $\Delta\alpha$ will concentrate in that corner, allowing the finite bending $\Delta\phi$ to spread over it, so that the bending energy, which depends on the ratio $\Delta\phi/\Delta\alpha$, is not excessively large.

When the curve is not given, then E_s should be minimized over all functions $\vec{t}(\alpha)$ and $l(\alpha)$, with all possible parametrizations α, which are consistent with the available data.

The analogy to a spring is not perfect, though, as an actual spring usually does not have a dimensionless E.

The normalization factor L is important in two ways: (i) it makes E, and hence the whole theory, dimensionless. Thus the resulting curve will be independent of the scale of the problem, which should not affect the curve fitting. This is unlike most other theories. (ii) In placing the knots, we are only interested in how much their distribution deviates

from a uniform one, and we want this deviation to affect the (stretching) energy. We do not want energy resulting from a uniform stretching of the spring to be counted. The normalization by the total arclength eliminates this contribution. The deprojection problem considered earlier can demonstrate this point. In that case, we are given an ellipse as the projected curve, and we want to find the space curve that most likely has given rise to this apparent ellipse. A planarity assumption restricts us to an ellipse with an unknown slant. Minimizing E, with the normalization factor, will yield a circle as the ellipse which minimizes the energy over the space of all possible deprojected ellipses, because it has the most uniform distribution of curvature (and of coil density). Without normalizing, however, the circle is *not* the minimizer because a lower energy state can be obtained by shortening the total arclength, even at the expense of deviating from uniformity. It is this contribution to the energy from change in the arclength that is eliminated by normalizing the arclength. The circle is the more desirable result as it is consistent with the other theories mentioned above, as well as with human perception.

The potential $V(\vec{x})$ is determined as follows: Let us assume that attached to each point $\vec{x}_d$ of the given data is an attractive potential $g = g(\vec{x} - \vec{x}_d)$, which represents the force that this data point exerts on a curve point lying at $\vec{x}$. Then, the influence of the entire given picture at each point $\vec{x}$ is a convolution of g with the picture f:

$$V(\vec{x}) = \int f(\vec{x}_d) g(\vec{x} - \vec{x}_d) d\vec{x}_d$$

For a discrete data set the integral is of course replaced by a summation.

In least squares fitting, g is proportional to r^2 (r being the distance $|\vec{x} - \vec{x}_d|$). A better choice is an f that decreases with distance such as a Gaussian, a Gaussian divided by r, or a Lorentzian function. The attraction force can be found by differentiating V.

We will now examine the behavior of E_s more precisely in various cases where the extremization can be done analytically, and show that it meets our demands.

For the interpolation problem, we have to consider curves which satisfy given boundary conditions, and V is unnecessary. If we assume $\kappa = 0$ in E, we obtain the case discussed in the previous section. In the general case, we can again use the EL equations. We will now have two EL equations from which we can determine both these functions, and thus find both the curve function $\vec{v}(l)$ and the new coordinate $\alpha(l)$. The two EL equations are very simple and have the solution

$$l = \alpha$$
$$\phi = (\phi_2 - \phi_1)\,\alpha + \phi_1$$

Substituting l for α in the second equation we immediately obtain the circular arc we had before. Thus, our new modifications at least did not destroy the desirable results we have had before with E_n.

We shall first examine the behavior of E_s in a simple curve containing small and large scale variations, and compare it to the behavior of the former energy E_n. This curve is a simple isolated corner. In the next section we shall use the results obtained here for handling a full shape containing corners, namely a parallelogram.

Consider two straight segments of length Δl, joined by a small circular arc of length $\Delta l'$, and making an angle $\Delta\phi$ with each other (Fig. 1). It is trivial to prove from the EL equations that on a straight line l must be proportional to α, and we have already proven that on a circular arc, l, α, ϕ are all proportional to each other. Thus, substituting these results in the general expression for E_s, the integration over the separate sections can be performed explicitly and E_s can be expressed by the finite increments in l,ϕ, α, as follows:

$$E_s = 2\frac{\Delta l^2}{\Delta\alpha} + \frac{\Delta l'^2}{\Delta\alpha'} + \frac{\Delta\phi^2}{\Delta\alpha'}$$

where $\Delta\alpha$, $\Delta\alpha'$ are the increments of α over Δl, $\Delta l'$ respectively. We want to extremize E_s with respect to α_i, subject to the constraint

$$2\Delta\alpha + \Delta\alpha' = 1$$

An elementary calculation shows that the extremum is obtained with

$$\Delta\alpha = \frac{\Delta l}{2\Delta l + R} \qquad \Delta\alpha' = \frac{R}{2\Delta l + R}$$

where $R = \sqrt{\Delta l'^2 + \Delta\phi^2}$.

Substituting these quantities in E_s we obtain in the extremum

$$E_s = (2\Delta l + R)^2$$

For a small corner, $\Delta l' \ll \Delta\phi$ and $\Delta l \approx 1/2$. We thus find the simple result

$$E_s \approx (1 + \Delta\phi)^2$$

This expression is independent of the exact shape and size of the corner, and depends only on the total turn $\Delta\phi$. The corner region Δl can be as small as we like without adversely affecting the energy E_s.

In comparison, the simple curvature minimization principle will give for this corner

$$E_n = \frac{\Delta\phi^2}{\Delta l'}$$

which tends to infinity as the length of the corner $\Delta l'$ goes to zero. Although we have dealt with an isolated corner, similar methods can be applied in more general cases, one of which is treated in the next section.

In conclusion, we have seen the different roles of the two terms in E_s. The first term will be rather dominant in sections of the curves having rapid curvature changes over short length, such as bumps, corners, or saw-teeth. The second term will be the major contributor in the large-scale features, having slow variations in curvature, such as the other boundaries of the saw. Thus we have obtained a "natural" way of treating curves with two very different scales of curvature change, without having to use arbitrarily preset filtering widths, as is commonly done.

One may refine the way the arclength is normalized. The normalization to a total length of 1 is not essential. One can define $l = \nu s/L$, where ν is an arbitrary positive constant. This results in multiplying the second term in E_s, representing the bending energy, by ν^2. In this way one can control the relative contributions of the bending and stretching energies of the spring (instead of κ). The circular case, however, will not be affected.

IV. SKEW-SYMMETRY

As an application of the ability of our new measure E_s to handle sharp corners, one can consider the deprojection of a parallelogram. If we regard this curve as a planar shape which is slanted and tilted in space, the question arises what is the most likely shape that has given rise to this apparent parallelogram. According to our minimum principle, the most likely shape is the one that minimizes the spring energy E_s

Not surprisingly the result turns out to be a rectangle. This agrees with the maximum compactness method, the maximum likelihood method and the human observation "method". The first two, however, have very restricted domains of applicability.

It is only necessary to prove that a rectangle extremizes our energy E_s over the set of parallelograms, i.e. that the derivative of the rectangle's energy with respect to the unknowns vanishes. The unknowns here are the distribution of α along the curve, and the angle by which the parallelogram deviates from a rectangle. This can be proven using the previous result [Weiss, 1986a].

V. GENERAL SURFACES IN 3-D

So far we have dealt with planar curve, described by one coordinate. We now turn to the general case of a (reasonably) arbitrary surface in 3-D. Such a surface can be parametrized by two coordinates α_1, α_2. Our energy E_s will now be a double integral:

$$E_s = \int_0^1 \int_0^1 \sum_{i=1}^{2} \left[\left(\frac{d\vec{t}_i}{d\alpha_i} \right)^2 + \left(\frac{dl_i}{d\alpha_i} \right)^2 + V \right] d\alpha_1 d\alpha_2$$

where $\vec{t}_i$ and l_i are the normalized length and the tangent, respectively, of the curve parametrized by α_i.

Returning to our "spring" model, one can visualize the surface as made up of a grid of springs, which form its natural coordinates, with α_i being the number of coils between a surface point and the boundary, along a coordinate.

Given a set of known contours, either on the boundary or otherwise, and suitable information about the surface normal on them, we reconstruct the surface that best fits these contours as the one that extremizes the above integral (over all the surfaces that comply with the data). The α_i will now have a more visual meaning then in the one-parameter case: they will be a pair of natural coordinates parametrizing the surface. Although general experiments have not been conducted, in several important cases, such as planar geodesics discussed later, these curves seem to be the ones that our visual intuition would expect as natural, and so we refer to them as "natural coordinates". This is another advantage of the new procedure over the simpler energy principle. It is interesting that the variables α_i can serve the dual purpose of both handling bumps (and straight lines) and providing a set of 3-D natural coordinates.

A simple example is that of circular boundary which is assumed to be occluding. It is a common assumption that for a smooth surface, the surface normal along an occluding contour lies in the projection plane. It is easy to prove that the this boundary will give rise to a (hemi-)sphere, complete with its longitudinal and latitudinal coordinates, which will be represented by α_1 and α_2. Rather than do that, we shall prove a general theorem about curves on surfaces, which will be very useful in finding natural coordinates as well as the surfaces themselves (including the sphere). Of course, for generic data the minimization can only be done numerically.

We first define a *curve of local symmetry* Γ as one which divides the surface into two parts that are symmetric in the vicinity of the curve. Put another way, a reflection, say of the surface lying to right side of Γ, will match the left side, near Γ. An example is the center line of any fold, ridge, valley or corrugation on a surface (Fig 2), if this line is planar. Another example is the three symmetry lines on an ellipsoid (this is a global symmetry).

An alternative definition is that the curvature in the direction perpendicular to the curve has an extremum at each point along the curve.

Theorem 1: A curve of local symmetry is planar.

Proof: Trivial. A reflection of the right side of Γ will not match the left side unless Γ is planar (Fig 2).

Theorem 2: A local network of coordinates, consisting of a local symmetry curve Γ and curves that are orthogonal to it, locally extremizes the energy E_s.

By "locally extremize" we mean that an infinitesimal variation of this coordinate patch in the vicinity of Γ will leave E_s unchanged (i.e. $\delta E_s = 0$). The proof is in [Weiss, 1986a].

A very interesting class of surfaces to which our theorem is easily applicable is the class having an "ignorable coordinate", namely, its curvatures do not change as we move along this coordinate. As sub-classes one can mention (a) strips, or pipes, of arbitrary cross section, whether straight, circularly or helically bent (Fig 3), in which the ignorable coordinate is the length of the pipe, and (b) surfaces of revolution, in which case the azimuth ϕ is "ignorable" (Fig 2a). Any coordinate line *orthogonal* to the ignorable direction can be regarded as a local symmetry curve, as the surface stays the same on its sides. Thus, this curve, and the curves in the ignorable direction that intersect it, form a local network of curves that satisfy the conditions of Theorem 2, and thus it locally extremizes E_s.

A further consequence of Theorem 2 is the ability to make predictions about surfaces when they are not known

in advance, without having to actually extremize E_s or solve the EL equations. It is clear that generally, the greater the number of extremal local coordinate patches a surface has, the more "extremal" the global 2-dimensional integral E_s will be. (This would be clearer if this "extremum" were a "minimum", which it usually is, but as we have not examined here the second derivatives, we shall stick to the term "extremum".) Thus, a surface that extremizes E_s will contain as many symmetry curves as are compatible with the initial data. As a consequence, we will tend to obtain surfaces containing planar, spherical or cylindrical parts, exact or approximate, rather than bumpy ones. As a particular example, we can use this general result to conclude, without having to solve the EL equations, that a circular occluding boundary will give rise to a sphere. This is because a sphere is the most symmetric surface (consistent with this boundary condition), thus having the greatest quantity of symmetry curves. More generally, boundary contours such as those of Fig. 2a will give rise to a surface of revolution, because this surface consists of a collection of symmetric local networks (the meridians plus the parallels) which fits those contours. Similar arguments apply for other surfaces.

It is reasonable to assume that when the conditions of the theorem are satisfied only approximately, e.g. when the curve is only approximately symmetric, a similar parametrization will still take place, with the natural coordinates approximately following the quasi-symmetry curves. Thus our theory is applicable to shapes such as generalized cones, as long as the flutings vary slowly on the length scale of the cone's radius. This is also consistent with human perception (Fig. 4).

As in the 2-D case, one can incorporate into the reconstruction data from curves and points not lying on the surface, by adding to E_s terms representing an attraction energy between the surface and this data.

VI. DISCUSSION AND RELATION TO OTHER WORK

In the preceding sections we have suggested a new shape reconstruction method, with varying mesh, and solved it analytically in a few cases. These cases were solvable mainly because of their symmetry. The notion of symmetry plays a central role in other work as well. We have already mentioned the Brady-Yuille compactness measure, used for the deprojection of closed planar curves. As compactness, or the ratio of area to the perimeter squared, is generally higher for more symmetric shapes, the compactness can be viewed as a good measure of global symmetry (as distinct from our local symmetry).

Brady & Asada [1984] have studied local symmetries in planar shapes. Strictly speaking, our definition of "local" applies to an infinitesimal vicinity around a curve. In this sense, every straight line on a plane is a local symmetry curve, except at its edges. Brady examined wider vicinities, that extend to the nearby boundaries of the curve, so we shall elevate the type of symmetry he treated to a "regional" symmetry. Brady has shown that this kind of symmetry is very useful in representing and handling many types of shapes. This indicates that symmetry is of fundamental importance at all spatial scales (global, regional, and local) of shape representation. We used symmetry in an analytic treatment of shapes in which it strictly holds, but its application is probably much wider: symmetric shapes can be used as a first approximation in a numerical computation of more general shapes. This will potentially increase the computational efficiency in reconstructing shapes which possess some axial symmetry, such as "generalized cylinders", or that can be decomposed into such shapes.

In differential geometry terms, our local symmetry curves are planar geodesic ones. Brady *et al.* [1984] and Stevens [1981] have shown that planar geodesics are usually "natural" coordinates. However, not all natural coordinates are planar geodesics. For example, the parallels on a surface of revolution, and the spirals on a helicoid (Fig. 5), are not planar geodesics [DoCarmo, 1976]. The question why these look natural was left open, as they did not seem to fit any consistent rule. In our theory, these curves are natural coordinates by virtue of their being *orthogonal at each point of their length to planar geodesics*, namely the meridians and the rulings, respectively. This one rule fits all the above cases. Moreover, unlike the previous work, our results are part of a general theory of shape representation derived from sound first principles.

The above mentioned papers contain many numerical techniques for finding symmetries, geodesics, etc. Their agreement with our work suggests that many of these techniques will be very useful in a computational implementation of our theory.

Although we have only examined reconstruction from spatial data, such as points and curves, other types of information can be included in the energy functional to be minimized. It was shown [Terzopoulos, 1985] that shading, stereo and other data can be cast in an energy form and treated as a variational problem. Constraints suplied by a knowledge- based system can also be added. For example, one can add a term to the energy measuring how close the shape is to a given template. Investigations of these possibilities are currently under way.

VII. NUMERICAL EXPERIMENTS: INTRODUCTION

In the following sections we will describe our numerical experiments with the method. A subset of the surface interpolation problem is the fitting of *curves* to given data *points*, either in the plane or in 3-D space. This subset retains many of the important properties of the full 3-D problem. We have implemented this restricted problem numerically. Thus, we have been able to investigate many of the major issues of the numerical implementation of the full 3-D problem, with the program being much easier to develop and much faster to run. Some of the results and lessons learned from this implementation are presented in this report. A subsequent report will contain the full 3-D implementation.

The 2-D implementation has merits in it's own right. It defines a new method of interpolation in which the mesh size is automatically and optimally adapted to the local

variation of the curve, with a finer mesh being used in rapidly changing sections of the curve.

Our experience with the numerical implementation has led us to make a number of refinements to this basic model, that do not change the basic principles. The main ones are:

• The use of the normalization factor makes the problem a global one, as a change in one part of a curve changes the energy everywhere, through L. This is undesirable, especially if parallel computation is to be implemented. One can circumvent the problem by treating L as an independent variable, with the normalization being imposed as a constraint, which is added to the energy with a Lagrange multiplier λ. Thus, a fourth term is added to the energy, in the form

$$\frac{\lambda}{L}\int_{\alpha_0}^{\alpha_1}\frac{ds}{d\alpha}d\alpha-(\alpha_1-\alpha_0)=0$$

The constant $\alpha_1-\alpha_0$ does not need to be added to the functional E, because the minimization will not change it, while the integral part joins the other integrals in E. E has now to be minimized with respect to L, as well as with respect to the curve, and the above constraint equation has to be satisfied. These added two equations enable us to determine L and λ. The problem is still global, but the global calculations have been isolated to these last two equations, and they do not need to be done for each knot point, which is a great relief.

It can be shown that for a small attraction potential, and a nearly uniform knot distribution, the constraint equation is approximately satisfied if $\lambda\approx-2$. Thus, by substituting this value, the amount of global calculation is further reduced.

• As $\vec{t}=d\vec{x}/ds$, the bending term can be written explicitly as

$$\left(\frac{d^2\vec{x}}{d\alpha ds}\right)^2$$

¿From the point of view of the basic principles, other forms of dependence of $\vec{x}$ on α, s are acceptable. For instance, one can reverse their order. These variables are not independent and the exchange will change the value of the bending term. One can prove that

$$\left(\frac{d^2\vec{x}}{dsd\alpha}\right)^2=\left(\frac{d\alpha}{ds}\right)^2\left(\frac{d^2\vec{x}}{d\alpha^2}\right)$$
$$=\left(\frac{d^2\vec{x}}{d\alpha ds}\right)^2+\left(\frac{d\vec{x}}{d\alpha}\frac{d^2\vec{x}}{d\alpha^2}\right)^2\left(\frac{d\alpha}{ds}\right)^4$$

A linear combination of the two possibilities is also acceptable. The form in the first line above has proven preferable numerically. Furthermore, one can get rid of s altogether in the bending term, and define the curvature purely as a function of α. By dropping the factor involving $d\alpha/ds$ above, the curvature becomes simply $\frac{d^2\vec{x}}{d\alpha^2}$, with its square being the bending energy. This will strengthen the effect of concentration of points in corners, as we shall see.

VIII. CHOICE OF CURVE REPRESENTATION

So far we have dealt with idealized curves having an infinite number of degrees of freedom. For numerical implementation we have to choose a finite representation. Several questions arise, which do not arise in standard interpolation schemes. Normally, one has knot points at fixed positions along the x axis, so that the x values are known and one seeks the $y(x)$ values. In our case the knot points are distributed along the the curve in accordance with the parameter α, which is itself unknown. Both x and y are functions of α. Under these conditions, how shall we construct the mesh and how shall we perform the discretization? What shall we choose as independent variables and which are the dependent ones? Additionally, what is the best way to calculate the derivative with respect to α?

The knot points have been handled as follows: Our unknowns are the x,y coordinates of n knot points, namely $\vec{x}_1,\ldots,\vec{x}_n$, ($2n$ variables). More accurately, as we shall see, we use an equivalent linear combination of these values as unknowns. These unknown points are assumed to be equally spaced with respect to the α parameter, with the interval length $\Delta\alpha$ given, $\Delta\alpha=\frac{(\alpha_1-\alpha_0)}{(n-1)}$. We thus obtain E as a function of the unknowns $\vec{x}_i$. This has turned the problem into a minimization of $E(\vec{x}_i)$ with respect to $\vec{x}_i$.

Given $x_i(\alpha), y_i(\alpha)$, the functions $x(\alpha), y(\alpha)$ and their derivatives with respect to α can be approximated in several ways. The particular way used can have a profound effect on the stability and convergence of the algorithm. While we have not done a thorough comparative study of the merits of the various methods, we have considered the main approaches. We used the general guidance of the literature to narrow down our choice, then tried two methods for a while until we settled on one.

First, we have the choice between the finite difference method and the finite element method. The former is based on a straightforward substitution of each derivative by its finite difference analog, e.g. $dx/d\alpha$ is replaced by $\Delta x/\Delta\alpha$. The method has the advantage of a simple implementation. However, it has a greater potential for instabilities then other methods, and it does not give the value of the curve *between* the knot points, which makes it inconvenient for an interpolation scheme. So we opted for the finite element method. This method consists of approximating the function as a superposition of (usually local) basis functions. These basis functions, which are typically piecewise polynomials, can easily be differentiated so that the derivative of the function at each curve point can be obtained. (We have two separate functions here: $x(\alpha), y(\alpha)$.) Through the minimization, one can find the coefficients of the superposition. These are related through a known linear transformation to the values of the functions at the knot points, $\vec{x}_i$, which are usually of more interest.

Now one has to choose appropriate basis functions. The simplest possibility is using triangular ("roof") basis functions. This is equivalent to representing the curve by a chain of line segments. One can use the slopes of the

lines $((x_i - x_{i-1})/\Delta\alpha, (y_i - y_{i-1})/\Delta\alpha)$ as the first derivatives, while the angle between adjacent segments is related to the second derivative, or curvature. Thus we have all the derivatives we need as functions of the unknowns $\vec{x}_i$ and the implementation is quite straightforward in this respect. The decomposition into basis functions does not need to be done explicitly in this case. The superposition coefficients are simply the $\vec{x}_i$, the values of the function at the knot points. (The first derivatives, but not the second ones, are the same here as in finite difference method).

This method was tried for a while, and we encountered some problems. In certain situations, particularly around corners, which is the case in which we are most interested, the knot points are quite close to each other, as expected. The line segments connecting them are quite short. For a very short segment, any small movement in the positions of its ends causes a considerable change in its inclination. In other words, when the length of the line segment tends to 0, the derivative $dE/d\vec{x}_i$ tends to infinity. This is easily seen mathematically, as this derivative contains a factor of $1/\Delta s_i$ (Δs_i being the length of the line segment.) This large derivative can cause a problem for the minimization technique, which needs these derivatives. It also means that the curve can become unstable around corners, with its exact shape affected by small errors in $\vec{x}_i$. This is especially noticeable with the limited accuracy of 32-bit single precision arithmetic.

The situation is not hopeless. It has been alleviated by working in double precision, and by limiting the segments' length. Also, the derivatives $dE/d\vec{x}_i$ have been computed analytically rather then numerically. Nevertheless, we found it more convenient to work with a different scheme, which is inherently more stable in this respect, namely choosing cubic splines as our basis functions.

Here again one has a choice. One possibility is to use the cubic Hermite polynomials [Strang and Fix, 1973]. These basis functions are defined on a very small local support, namely one that consists of the two mesh intervals $[\alpha_{i-1}, \alpha_{i+1}]$ around a mesh point with α_i, and they vanish outside of this domain. There are two independent basis function attached to each knot point, and the coefficients multiplying them are linearly related to the value of the function and its first derivative at that knot point. The basis function thus depends on the local properties of the curve only. In all, we have $2n$ unknowns for $x(\alpha)$, and similarly for $y(\alpha)$. Representing the curve by Hermite cubic polynomials would be relatively simple because of the local nature of the superposition, but we would have to minimize E with respect to both $\vec{x}_i$ and the derivatives $d\vec{x}_i/d\alpha$.

One way to deal with the unknown derivatives is to approximate them by the known values of the function in the vicinity, using a finite difference-like method. The Bessel and the Akima cubic splines [DeBoor, 1978] are obtained this way. We have not tried these methods but we feel that they do deserve a fair chance.

The superposition of Hermite polynomials is continuous and has a continuous first derivative. (This is because the basis functions and their derivatives vanish at the ends of the intervals they are supported on.) One can reduce the number of unknowns by requiring higher order continuity, e.g. a continuous second derivative. A price is paid by reducing the locality of the representation. In this technique the curve is represented as a superposition of cubic B-splines. These basis function are piecewise cubic polynomials supported on a larger interval $[\alpha_{i-2}, \alpha_{i+2}]$ around the knot point. There is only one possible basis function at each knot point, reducing the number of coefficients, and thus of unknowns, to n for each component (x, y). The relation between the superposition coefficients and the curve function values is no longer local. At each knot point, a contribution to the superposition arises from the basis functions attached to the two neighboring knots. This makes the calculation at each point more complex and time consuming. In our evaluation, though, this is more than offset by the reduction in the number of unknowns. Thus we eventually decided to use cubic B-splines as the basis functions for our curve components. These functions have vanishing values, first, and second derivative at the interval edges and thus the superposition is twice differentiable. Another incentive is the inherent relation between these splines and minimal curvature. It can be shown that among all twice differentiable functions which pass through known data points, a piecewise cubic polynomial has the lowest overall curvature. This statement refers to the functions $x(\alpha), y(\alpha)$. The implication for the function $y(x)$ is not immediately clear.

The next section contains a more precise mathematical description of our spline representation.

IX. THE SPLINE REPRESENTATION

A major distinction between our representation and most other spline fittings of a curve is that in our case x and y are both piecewise polynomials in some variable α, rather then y being a polynomial in x. This has advantages in representing non-polynomial curves, such as circles, for which splines a are notoriously poor approximation. In our representation the circle can be described by the coordinates $y = \cos\alpha$, $x = \sin\alpha$. The sin and cos functions are very well approximated by polynomials (their Taylor expansions, for instance), and thus the splines are a good approximation. This is in contrast to the circle being represented as $y = \sqrt{1 - x^2}$, which is hard to approximate by polynomials. The same goes for other conic sections. We stress that α is not the length between data points as in some applications, but an unknown function of x, y to be found by a minimization process.

Denoting the basis function at knot point i by $B_i(\alpha)$, and its (two valued) coefficient by $\vec{v}_i$, the curve can be written as

$$\vec{x}(\alpha) = \sum_{i=0}^{n+1} \vec{v}_i B_i(\alpha)$$

Based on the form of the B_i, the functions $\vec{x}$ can be calculated at each interval $[\vec{x}_i, \vec{x}_{i+1}]$ as follows [DeBoor, 1978, Barnhill and Riesenfeld, 1974]. Define on each interval a parameter $t = (\alpha - \alpha_i)/(\alpha_{i+1} - \alpha_i)$, running from 0 to

1 proportionally to α. We now have curve segments $\vec{x}_i(t)$ with

$$\vec{x}_i(0) = \vec{x}_i$$
$$\vec{x}_i(1) = \vec{x}_{i+1}$$

On each interval the curve is (for knot points equidistant in α)

$$\vec{x}_i(t) = (t^3, t^2, t, 1)(C)(\vec{v}_{i-1}, \vec{v}_i, \vec{v}_{i+1}, \vec{v}_{i+2})^T$$

where (C) is the matrix

$$(C) = \frac{1}{6}\begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{pmatrix}$$

We shall also need the first and second derivatives, which are easily obtained from the above equations, giving

$$\frac{d\vec{x}_i(\alpha)}{d\alpha} = \frac{1}{\Delta\alpha}(3t^2, 2t, 1, 0)(C)(\vec{v}_{i-1}, \vec{v}_i, \vec{v}_{i+1}, \vec{v}_{i+2})^T$$
$$\frac{d^2\vec{x}_i(\alpha)}{d\alpha^2} = \frac{1}{\Delta\alpha^2}(6t, 2, 0, 0)(C)(\vec{v}_{i-1}, \vec{v}_i, \vec{v}_{i+1}, \vec{v}_{i+2})^T$$

Another quantity needed in E is

$$\frac{ds}{d\alpha} = \sqrt{\left(\frac{dx}{d\alpha}\right)^2 + \left(\frac{dy}{d\alpha}\right)^2}$$

One can see that knowing the coefficients $\vec{v}_i$ is equivalent to knowing the knot points $\vec{x}_i$. The formula above makes the linear transformation from $\vec{v}_i$ to $\vec{x}_i$ immediate. The reverse calculation, from $\vec{x}_i$ to $\vec{v}_i$, is less immediate as one needs to solve a (simple tridiagonal) system of n linear equation. Because of this equivalence we can use the coefficients $\vec{v}_i$ as unknowns rather then the $\vec{x}_i$, which eliminates the need for solving that system.

The coefficients $\vec{v}_i$ have an important geometric interpretation. They can be regarded as the coordinates of "guiding polygon" vertices. The curve generally follows this polygon, but is of course smoother and is guaranteed to have lower variation. So, the vertices' coordinates $\vec{v}_i$ are not very far, in general, from the knot points $\vec{x}_i$. In applications in which it is not important for the interpolating function to pass through the data points, one can use the data points as vertices of the guiding polygon, i.e. as the spline coefficients $\vec{v}_i$. Then, $\vec{x}(\alpha)$ and their derivatives are immediately computable by the above formula (assuming α is approximated, say, by s). This can be useful in obtaining a first approximation for the curve in the minimization process. In some applications, it is the knot points $\vec{x}_i$ which are given as data, and the curve is forced to pass through them. Then one must solve the above system of linear equations to find the coefficients $\vec{v}_i$. In our case this is unnecessary as we treat the data points entirely differently (next section).

It remains to deal with the boundary conditions. As the value of the function at each knot point depends on the contributions from the B-splines attached to the neighboring points, a problem arises at the edge points. Note that in the superposition above the limits are from 0 to $n+1$, with the knot points indexed by $i = 1, \ldots, n$ only. One needs an extra assumption for determining $\vec{v}_0, \vec{v}_{i+1}$. This can be regarded as extending the guiding polygon in some reasonable way. A common assumption is that the curvature at the edges vanishes. This can be approximated by requiring a vanishing second derivative. ¿From the expressions above for the derivatives it is easy to prove that under this assumption

$$\vec{v}_0 = 2\vec{v}_1 - \vec{v}_2$$
$$\vec{v}_{n+1} = 2\vec{v}_n - \vec{v}_{n-1}$$

We found it more suitable to assume a *constant* curvature at the ends, rather then zero. This is especially applicable to circles, or circular arcs, which are a part of many curves, at least in small neighborhoods. This assumption means a zero third derivative, which leads to

$$\vec{v}_0 = 3\vec{v}_1 - 3\vec{v}_2 + \vec{v}_3$$
$$\vec{v}_{n+1} = \vec{v}_{n-2} - 3\vec{v}_{n-1} + 3\vec{v}_n$$

X. REPRESENTING THE DATA

Rather then forcing the curve to pass through the data points, we assign to each such point an attraction force which it exerts on the curve. This force should possess some desirable characteristics. It should be strong in the vicinity of the point and fall off at a large distance. The usual least squares measure has the opposite property. It emphasizes the effects of far away points, which have little to do with the data to be interpolated. The vicinity in which the force is most effective should be controllable, as well as the strength. This will reflect the degree of reliability in determining the position of the data point. A reliable data point can be assigned a strong force in a smaller vicinity. The force is proportional to the length of the curve on which it acts, ds, for it acts independently on each (infinitesimal) part of the curve. We found it necessary, in addition, to divide the force by the total length of the curve L, to eliminate the incentive for the curve to increase its length without getting closer to the data points. Without this normalization the curve may, for instance, oscillate around the data points.

The attraction force is thus represented in the total energy E by the potential *energy* $V(\vec{x})ds/L$. The potential *function* $V(\vec{x})$ is the cumulative influence of all the data points at the point $\vec{x}$, and is independent of the curve. The potential energy of the curve thus depends on the magnitude of the potential function V at the points through which the curve passes. The force itself is the derivative of V. The contribution of each data point to $V(\vec{x})$ can be chosen in several ways, consistent with the requirements above. Denoting this contribution by $g(r)$, where $r = |\vec{x} - \vec{x}_k|$ is the distance between the point $\vec{x}_i$ and a data point $\vec{x}_k$, one may choose

$$g(r) = \frac{ae^{-r^2/b^2}}{1+cr}$$

where a, b, c are parameters at our disposal. This potential falls off at large distances, with its effective range being determined by b; a is the strength of the contribution, and c helps in shaping the force near the data point. We found a Gaussian function divided by r (i.e. $c \gg 1$) more effective in some cases then a pure Gaussian ($c = 0$) although the difference is not crucial. Other functions are possible but have not been tried, such as as a Lorentzian resonance function

$$g(r) = \frac{a}{r^2 + b^2}$$

This too satisfies our requirement, without the need to calculate exponential functions, which may be time consuming on some machines.

The potential at a point $\vec{x}$, $V(\vec{x})$, can be regarded as a discrete convolution over the picture f:

$$V(\vec{x}) = \sum_n f(\vec{x}_n) g(\vec{x} - \vec{x}_n)$$

where the summation is over all pixels of the picture, and $f(\vec{x}_n)$ is the picture intensity at the pixel at $\vec{x}_n$. In the case of a simple set of data points to be interpolated, f equals 1 at a data point and 0 elsewhere. In that case $V(\vec{x})$ is simply the sum over the data points $\sum_k g(\vec{x} - \vec{x}_k)$.

Now the question arises how best to calculate V. V has to be known at every point where the curve is calculated, rather then at every picture point. Thus the convolution need only be done at points lying on the curve, and this can save a considerable amount of time, even if several curves are computed as iterations in the minimization process.

In fact even with this saving the convolution has proven rather time consuming on a VAX, and we used a different approach. We calculated the convolution on a fixed rectangular grid, then interpolated V at the curve points using the V values on these fixed grid points. This amounts to fewer points on which the convolution is calculated, because the number of iterations is usually quite large. Typically we used a 20×20 grid, while the curve had 20 points and was iterated 70 times. The interpolation was carried out using the cubic Hermite splines mentioned earlier, in a 2-variable generalization.

Now that we can express the both curve and V at every point, we can calculate the integral E. The integration has to be done numerically on each interval $[\vec{x}_i, \vec{x}_{i+1}]$. Since the integrand is made up of polynomials with a limited number of parameters, only a few internal points are needed in each interval. In fact we found that using the mid-point, with Simpson's rule, is a satisfactory approximation. (In spite of the polynomial representation, the integral cannot be performed analytically due to the presence of ds which involves a square root, and even otherwise it would be quite tedious.)

XI. THE MINIMIZATION METHOD

Once we have expressed the energy E as a function of a finite number of unknowns, x_i, y_j, it remains to minimize it with respect to these unknowns. More decisions are needed here. Basically there are two avenues open before us. The first is to find the minimum point by solving the system of non-linear equations

$$\frac{dE(x_i, y_j)}{dx_i} = 0$$

$$\frac{dE(x_i, y_j)}{dy_j} = 0$$

In the theoretical paper [Weiss, 1986a] we used the Euler-Lagrange equations to minimize E analytically in some simple cases. The above minimization method, where the quantities in E are expressed as a superposition of finite elements, amounts to discretizing the Euler-Lagrange equation by the Rayleigh-Ritz method.

The best routine we know of for the solution of a non-linear system of equations is in the MINPACK library, available from Argonne National Laboratory. It also appears, under the name SNSQE, in the National Bureau of Standards library and the SLATEC library. It is based on a modified Newton method, combined with Powell's method.

This method was on the whole not successful in our application. The Newton method needs a pretty good initial guess in order for it to converge. Except in special cases where we could provide one, the method rarely converged, and when it did it was not to a minimum but to some stationary point. Thus, it wasn't hard to decide to take the other route, and treat the problem as a non-linear optimization problem.

There are still several choices. We focused on two, the conjugate gradient method, and the variable metric BFGS method, both also available from Argonne National Laboratory. Both methods work well and always converge to a solution. The conjugate gradient method is guaranteed to converge to the minimum in n steps, where n is the number of unknowns, for a quadratic function E. Our problem is in general not quadratic, and 3-4 times that number is needed. (The program uses a Beale restart if more then n steps are executed.) While the BFGS algorithm is usually faster, it requires considerably more memory then the conjugate gradient method, on the order of n^2 numbers vs. n. While the differences may not be very important for a curve, they may be of crucial significance for surface interpolation, where n can be of the order of 100×100. For a curve with 20 knot points, the BFGS was about 30-40% faster, depending on the particular problem, than the conjugate gradient method, but required 1000 memory cells vs. 200. ($\approx n(n+7)/2$ vs. $\approx 5n$).

At this point there arises the question of the uniqueness of the solution. For a convex function E, uniqueness is assured. If a function is convex with respect to its arguments, then it has only one minimum. (This has nothing to do with the convexity of the curve itself.) Our E is in gen-

eral non-convex. One reason is that the potential function V is not convex. Thus one can arrive at a local minimum, and be ignorant of a lower minimum nearby. The choice of the parameters in the attraction force g has an impact on the convexity of V. Quadratic gs would always yield a quadratic sum V, and thus V would have a unique solution. We rejected this kind of force because of its bias towards distant points. A finite range force that falls of beyond an effective distance (b in our notation), such as a Gaussian, is quadratic in an area the size of b, and thus there is a unique minimum at areas of roughly this size. For a long range force, or large b, there will be only a few minima in the picture. However, the resolution of the data is lower, as g smoothes the picture by being convolved with the picture. In addition, the shallowness of the resulting minima mean a slow convergence rate. A short range force, with small b, leads to a curve that follows the data more accurately, but with a more bumpy function to minimize. Thus the parameter b can control the overall smoothness of the curve to some extent, along with the total number of knot points.

Another consequence of the non-linearity is the loss of the guaranteed stability that the linear finite element theory gives. Instability occurs when the discretized problem has a solution of its own which is very far from the continuous one (such as a solution that oscillates sharply between the mesh points). This is a quite common problem in finite difference methods. The finite element method that we are using is well understood in the linear case and insures stability and convergence: the solution of the discrete equation will approach that of the continuous one as n increases. The non-linear behavior is not that well understood. Our implementation had a minor problem which we currently regard as instability. The curve sometimes "folds" on itself and turns sharply by 180^0. The turning region has of course a high curvature, but it is highly localized and the discrete mesh manages to avoid it. A detailed inspection of the calculation led us to change the curvature term by either interchanging s and α or, better still, eliminate s altogether and express the curvature as $d^2\vec{x}/d\alpha^2$ (see Sec. II). This has indeed almost eliminated the folding problem, although more clarification of the situation is needed.

XII. RESULTS

As a test of the basic principles of our theory we compared the performance of our method against that of a simpler energy principle [Horn, 1977] in representative cases and tested the effect of our innovations. We also tested the effects of several key ingredients of the theory. For this purpose a synthetic data set was sufficient.

We first considered an interpolation problem, in which information is given only at the end-points of the curve, while the rest of the curve has to be interpolated. Here, $V = 0$. Fig. 6 the curve obtained in such a case. This curve might represent an occluded segment of a shape whose visible part is represented by the dashed lines. When both tangents were $90°$, we have obtained a semi-circle. Without the normalization factor, an ellipse was obtained. The end-points and the curve tangents there were maintained by a strong quadratic potential applied at the end-points and the points adjacent to them.

We next examined the effect of our reparametrization of the curve by α. This was done by using a right-angle corner as our data and testing how well the curve can follow it. The spring was attracted to the data points by a force represented by a potential V described earlier. In the following figures, crosses represent the data points, and diamonds are the knot points. Fig 7a shows the curve parametrized simply by the arclength s. The positions of the knot points were determined by the requirement of minimal bending + potential energy. (No stretching.) In Fig 7c the curvature, or bending energy, was measured with respect to the parameter α, which was an unknown found by the minimization algorithm. Fig 7b is an intermediate form of the curvature, using both s and α. (The curvatures in Figs. 7a,b,c were expressed, respectively, as

$$\frac{\partial^2\vec{x}}{ds^2}, \quad \frac{\partial^2\vec{x}}{dsd\alpha}, \quad \frac{\partial^2\vec{x}}{d\alpha^2}.$$

One can clearly see that the corner is followed much more closely when α is used in place of s, as more points are concentrated in the corner in this case. The distance between the knot points in the corner in Fig 7c is about a quarter of their distance along the straight sections. In Fig. 7a, in contrast, where conventional curvature is used, the distance between the knots remains constant, and one can observe the strong tendency of this simpler energy measure to round the corners as the only way to minimize the total curvature.

The particular representation of the curve also has some effect on how well it can follow the data. In our B-spline representation, s has a continuous second derivative with respect to α, and this limits the amount by which the distance between the knot points can vary along the curve. In the line segment representation this variation was more pronounced and the density of knot points in the corner was greater then in the present examples, so it deserves a further look in spite of some instability problems. A representation by Bessel or Akima cubic splines, which have continuous first derivative, also merits consideration.

Our next test involves interpolating sparse data points indicating some arbitrarily winding curve. The curve is neither convex nor closed. There was no need for any conditions on the boundary. Fig. 8a shows the interpolating curve with 20 knot points (which do not coincide with the data points). In Fig. 8b the resolution of the curve was reduced to 10 points. The curve is smoother, but follows the data less closely. A similar effect can be observed in Fig. 8c; this time the curve's resolution was kept high, while the effective range of the potential g was increased. That has the effect of smoothing the data itself.

Finally, a planar curve in 3-D was obtained from input consisting of two end-points with known coplanar tangents. The result is similar to Fig. 7c, except that it lies in the 3-D space.

XIII. CONCLUSIONS

In this paper we have presented a new method of reconstructing a 3-D surface from given contours. Some advantages of this mechanism are its ability to handle shapes containing both small-scale and large scale features, and dimensionlessness. These are among the requirements that any reconstruction mechanism will have to meet. The method is based on representing the surface as a network of springs, with each (natural) coordinate curve on the surface being analogous to one spring. The springs contribute energy from both their bending and stretching, and the optimal surface is obtained by minimizing the energy of the system over all possible spring networks which conform with the given data.

The springs' "coils" represent knots of a varying mesh, that is used to overcomes the problem of variations of different scales in different parts of the curve. The mesh is finer, or the coils are more densely concentrated, in parts of the curve that change more rapidly. The same minimum principle determines both the optimal placement of the knots and the shape of the curve (or surface) between the knots.

We have examined a few cases that can be handled analytically, mainly because of their symmetry. A surface of revolution, for instance, along with its meridians and parallels, could be reconstructed from appropriate boundary conditions. These cases also seem consistent with human perception.

We have implemented the theory numerically in the 2-D case. The implementation has produced results on several levels. At the higher level of evaluating the theory as a whole, we have a proof of the basic principle described above. The program has worked well in this respect. As we saw, the curve has no problem following sharp corners, by placing more knot points there. This problem of different rates of change of a curve, such as the difference between a corner and a straight line, has plagued all other curve fitting techniques and it seems that our method, in principle, solves the problem. In addition, our method does not impose the restrictions that many other methods do. In particular, the curve need not be convex, closed, nor single valued. There remain other problems that usually arise in minimization methods. The non-uniqueness of the solution means that the method will need some higher level guidance to choose the "right" solution. Efficiency is another issue, although it was not our goal here. It would be interesting to consider the question of whether a multi-resolution algorithm, which was successful in accelerating the convergence of a linear, fixed grid minimization problem [Terzopoulos, 1984], would be applicable to our case.

On another level, the implementation has improved our understanding of the theoretical problem and led to some changes. Among them were: (i) The bending term was generalized somewhat, to include various possibilities of expressing the curvature as a second derivative of $\vec{x}$ with respect to s and α. (ii) The treatment of the global factor L. By treating it as an independent variable to be determined by the Lagrange multiplier method we have been able to keep the global computations to a minimum. This will be important in an implementation on a parallel machine.

On the numerical level, we have learned about various numerical aspect of the implementation and found suitable ways of representing and discretizing the curve and the data. We have dealt with convergence and stability problems, and have examined the effects of various parameters in the theory. All this has great value in implementing the more general case of 3-D surface interpolation.

REFERENCES

Barnhill, R.E. and R.F. Riesenfeld [1974], *Computer Aided Geometric Design*, Academic Press, New York.

Barrow, H.G. and J.M. Tenenbaum [1981], "Interpreting line drawings as three-dimensional surfaces", *Artif. Intell.*, **17**, 75–117.

Brady, J.M. and H. Asada [1984], "Smoothed local symmetries and their implementation", A. I. Memo No. 757.

Brady, J.M., J. Ponce, A. Yuille, and H. Asada [1984], "Describing surfaces", Proc. 2nd. Symp. Rob. Res. (Kyoto).

Brady, J.M. and A. Yuille [1984], "An extremum principle for shape from contour", *IEEE Trans. Patt. Anal. & Mach. Int.*, PAMI-6.

Courant, R. and D. Hilbert [1953], *Methods of Mathematical Physics*, Vol. I, Interscience, London.

Davis, L., L. Janos and S. Dunn [1982], "Efficient recovery of shape from texture", Computer Vision Lab, University of Maryland, TR-1133.

DeBoor, C. [1978], *A Practical Guide to Splines*, Springer-Verlag, New York.

Do Carmo, M. [1976], *Differential Geometry of Curves and Surfaces*, Prentice-Hall, Englewood Cliffs, NJ.

Grimson, W.E.L. [1981], "The implicit constraint of the primal sketch", MIT A.I. Memo No. 663.

Horn, B.K.P. [1981]. "The curve of least energy", MIT A. I. Memo No. 612a.

Marr, D. [1982], *Vision*, Freeman, San Francisco.

Pavlidis, T. [1977], *Structural Pattern Recognition*, Springer-Verlag, New York.

Rosenfeld, A. and A.C. Kak [1982], *Digital Picture Processing*, 2nd edition, Academic Press, New York.

Stevens, K.A. [1981], "The visual interpretation of surface contours", *Artif. Intell.*, **17**, 47–73.

Strang, G., and G.J. Fix [1973], *An Analysis of the Finite Element Method*, Prentice-Hall, Englewood Cliffs, N.J.

Terzopoulos, D. [1982], "Multilevel reconstruction of visual surfaces: variational principles and finite element representation." MIT AI Memo 671. Reprinted in *Multiresolution Image Processing and Analysis*, A. Rosenfeld (ed.), Springer-Verlag, New York, 1984.

Terzopoulos, D. [1985], "Computing visible surface representations", MIT AI Memo No. 800.

Weiss, I. [1986a], "3-D Surface Representation by Contours", Computer Vision Lab, University of Maryland, CAR-TR-222.

Weiss, I. [1986b], "Curve Fitting with Optimal Mesh Point Placement", Computer Vision Lab, University of Maryland, CAR-TR-224.

Witkin, A. [1981], "Recovering surface shape and orientation from texture", *Artif. Intell.*, **17**, 17–45.

Young, D. and R. Gregory [1973], *A Survey of Numerical Mathematics*, vols. 1 and 2, Addison-Wesley, Reading, MA.

Acknowledgement

The author thanks J. Michael Brady of MIT's AI Lab (now at Oxford University) for very valuable discussions, comments and suggestions.

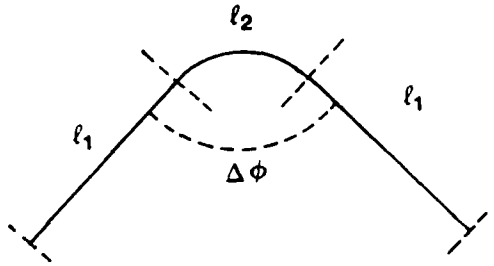


Fig. 1
A corner

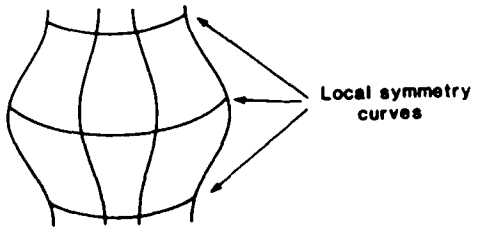


Fig. 2a

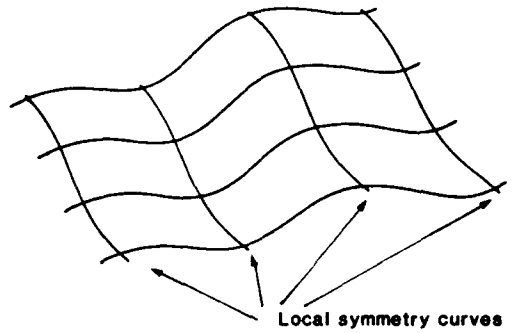


Fig. 2b
Local symmetry curves

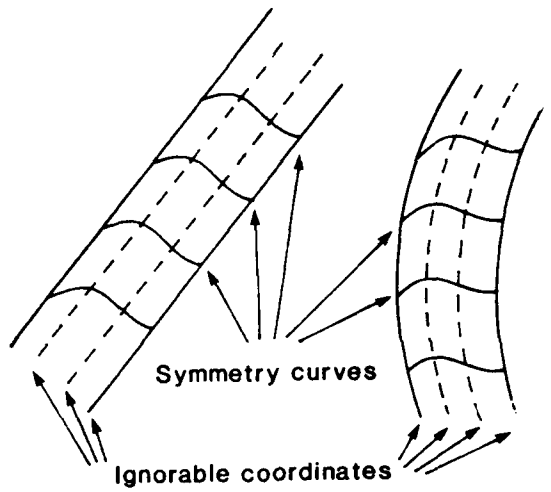


Fig. 3
Ribbons with symmetry curves

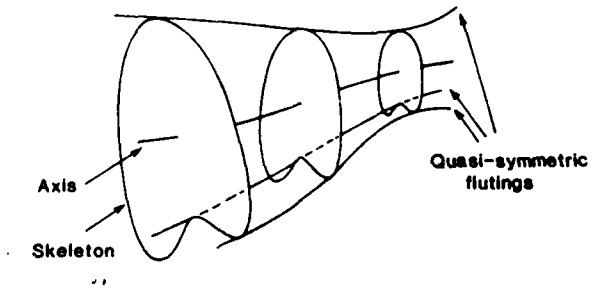


Fig. 4
Generalized cone

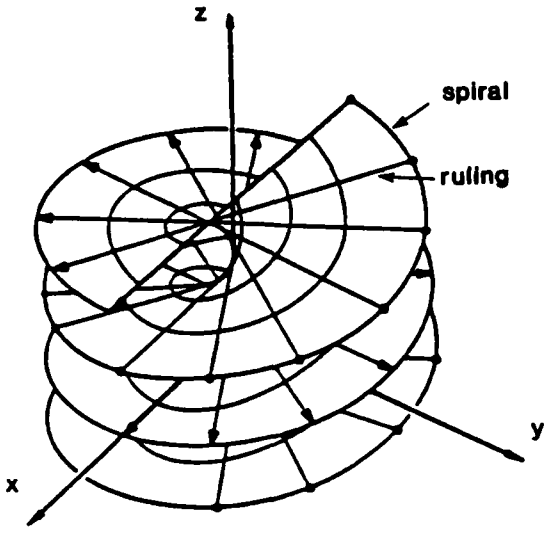


Fig. 5
The helicoid of a single blade
(Reproduced from do Carmo 1976, Figure 2–27, Page 94)

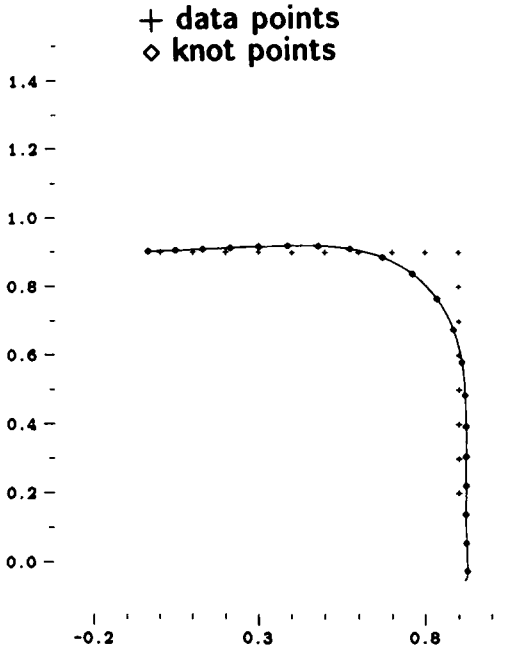


Fig. 7a

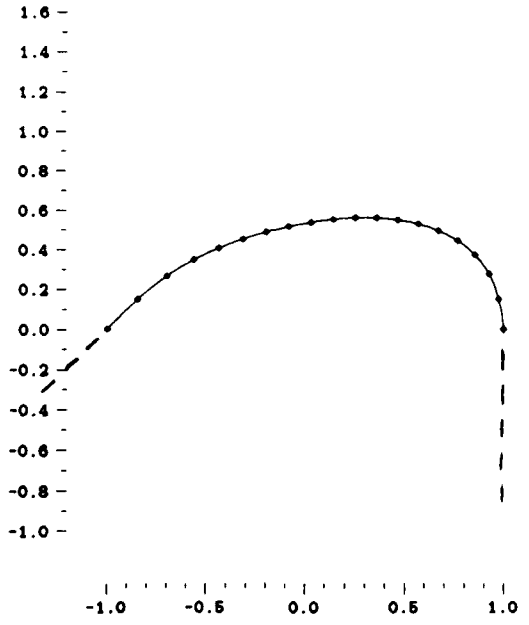


Fig. 6

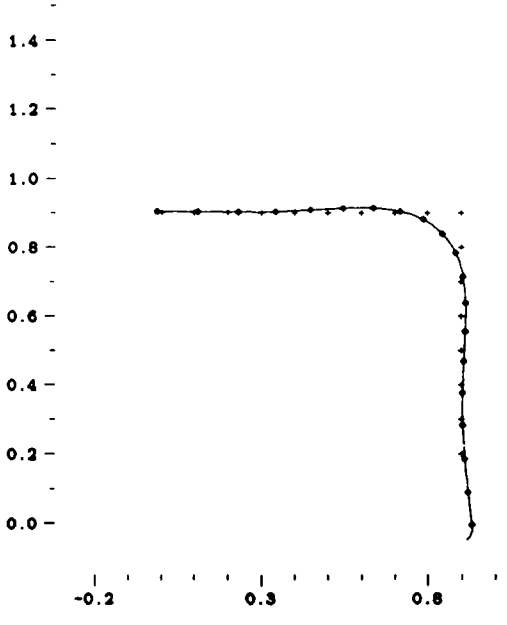


Fig. 7b

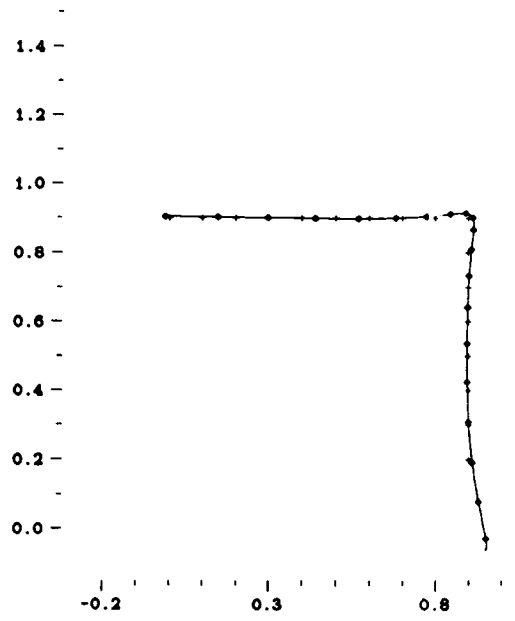


Fig. 7c

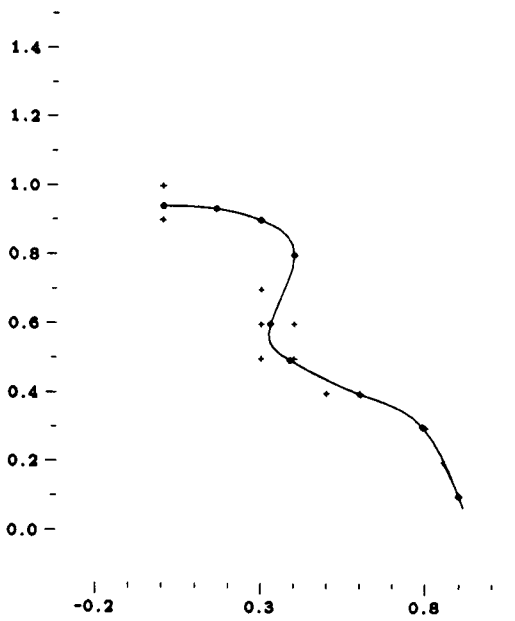


Fig. 8b

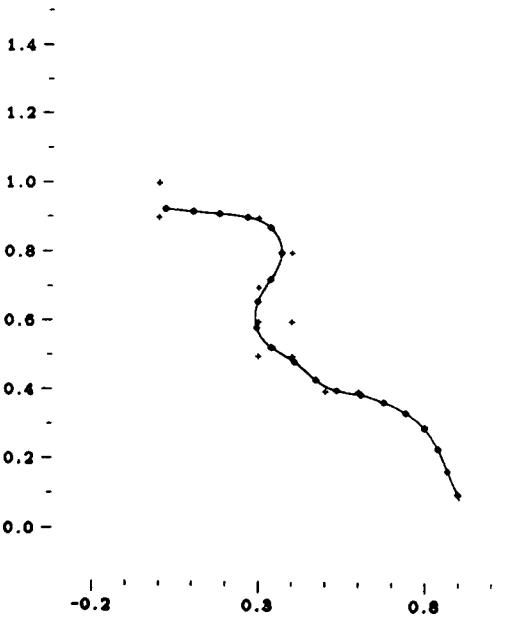


Fig. 8a

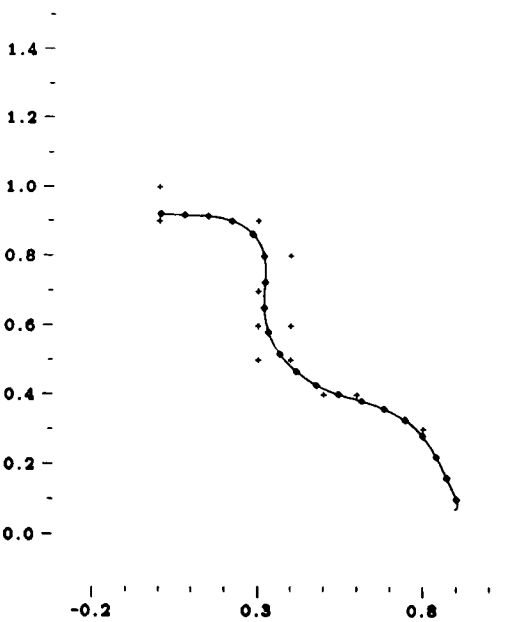


Fig. 8c

STEREO MATCHING USING VITERBI ALGORITHM

G.V.S. Raju*
Dept. of Electrical and
Computer Engineering
Ohio University
Athens, OH 45701

Thomas O. Binford
AI Laboratory
Computer Science Dept.
Stanford University
Stanford, CA 94305

S. Shekhar
AI Laboratory
Computer Science Dept.
Stanford University
Stanford, CA 94305

ABSTRACT

The key problem in stereo is a search problem which finds the correspondence between points in left and right images, so that, given the camera model, the depth can be computed from triangulation. This paper presents a stereo matching method using a modified Viterbi algorithm. Our algorithm matches surfaces between edges (scanline intervals between edgels belonging to edges) in the two images. Interval ratio, edgel orientation angles and contrast are used in developing a similarity function (cost or gain) to match edgels (intervals) in the two images on the same scanline. The major advantages of our method is in the computation. The method has been tested with different types of images and the results are presented in the paper.

INTRODUCTION

In the past few years, we have seen a growing interest in the application of 3D image processing. With increasing demand for 3D spatial information for tasks of passive navigation [1, 2], automatic surveillance [3], aerial cartography [4, 5], and inspection in industrial automation, the importance of effective stereo analysis has been made quite clear. A particular challenge in this area is to provide reliable and accurate depth data for input to object or terrain modeling systems.

Barnard and Fischler [6] viewed the problem of stereo analysis in terms of image acquisition, camera modeling, feature acquisition, image matching, depth determination and interpolation. The crucial one is that of image matching which identifies the corresponding points in two images that are cast by the same physical point in 3D space. This research work is concerned only with this problem. The paper addresses a research problem which finds the correspondence points between the left and right images, so that given the camera model, the depth can be computed by triangulation. Knowing the geometric relationships between the cameras used in the imaging can reduce image-to-image correspondence to a set of scanline-to-scanline correspondence problems. When a pair of stereo images is rectified, the epipolar lines are horizontal scanlines (rows). The epipolar lines are intersections of the two image planes with an epipolar plane which passes through an object point and the two camera foci. Figure 1 shows the geometry of this situation. In edge-based stereo, edges in the images are used as the elements whose correspondences are to be found.

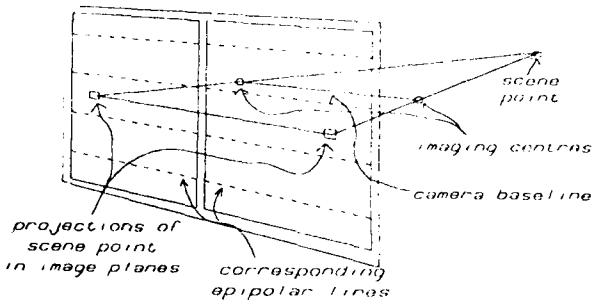


Fig. 1 Collinear Epipolar Geometry

A pair of corresponding edgels (edge-elements [7]) in the right and left images should be searched for only within the same horizontal scan lines. The search within the same horizontal scanlines can be treated as the problem of finding a matching path on a two-dimensional (2D) search plane whose vertical and horizontal axes are the right and left scanlines. A modified viterbi algorithm (a version of dynamic programming technique) [8, 9] can be used for this 2D search efficiently.

However, if there is an edge (curve composed of edgels) extending across scanlines, the correspondences in one scanline have strong dependency on the correspondences in the neighboring scanlines because if two edgels are on a non-horizontal edge in the left image, their corresponding edgels most likely lie on a corresponding edge in the right image [10]. To account for this, edge correspondence is sought through the modified viterbi algorithm.

The various stereo matching methods differ in the primitives used for matching, similarity measures used for matching, and the techniques (algorithms) used for local and global matching.

* On sabbatical leave at the Artificial Intelligence Laboratory, Computer Science Department, Stanford University, Stanford, CA 94305.

In this paper we use edgels as a primitive for local matching (scanline to scanline) and edges (curves composed of edgels) as a primitive for global matching. Several similarity functions are developed which are used to estimate the cost of edgel (interval) similarity. The stereo matching method developed in this paper also uses modified viterbi algorithm, as in the case of Arnold [8], Baker and Binford [9], and Ohta and Kanade [10], to determine edgel correspondence and edge correspondence. The major advantage of the method described here is in the computation.

First, we give a brief review of previous work in stereo matching, then we describe our method and the implementation details, and finally present some of the results of applying this method to real images and conclusions.

REVIEW

For stereo matching, area-based [11, 12] and feature-based [8, 9, 10, 13, 18] techniques have been used. In area-based analysis two-dimensional windowing operators measure the similarity in intensity pattern between local areas, or windows, in the two images. Cross-correlation is used to determine matches between windows in one image with windows in the other. Area-based correspondence has been applied quite successfully to the stereo analysis of rolling terrain, but it degrades when the scene is not smoothly varying and continuous. It requires the presence of detectable texture within each correlation window, and consequently it tends to fail in featureless, repetitive-texture environments, or surface discontinuities. At surface discontinuities there are occlusions and no possible correspondence between areas crossing the occlusion.

Feature-based systems match features derived from the two images rather than the intensity arrays. The most commonly used features are edges that are extracted from the intensity image for matching. The underlying principle is that a discontinuity in the intensity represents a discontinuity on the physical surfaces in the scene. The discontinuity can be due to surface depth, orientation, reflectance, or illumination. By matching these features we match physical curves on object surfaces.

The feature-based systems are less sensitive to photometric variations because they represent geometric properties of a scene and are faster since the number of features to consider are fewer than the number of pixels. Also, the match is more accurate as the features (edges) in an image can be located with subpixel accuracy. Since the feature matching leads only to a sparse depth map, we need to reconstruct dense depth map through surface interpolation, area matching, or model-based interpretation.

The most widely known edge-based stereo system is that of Marr and Poggio [14], as implemented by Grimson. It uses an uniqueness constraint to assign at most one disparity value to each point in the image and continuity requirement such that disparity values change smoothly, except at depth discontinuities.

Medioni and Nevatia [13] used segments, groups of collinear connected edge points, as matching primitives. In their study, the correspondence is based on a minimum differential disparity criterion. Arnold and Binford [15] developed an edge-based stereo correspondence system that used local edged properties to select edgel match possibilities, and a weighted iteration process to resolve match conflicts.

Baker [16], Arnold [8], Ohta and Kanade [10], and Baker and Binford [9] have used dynamic programming technique (viterbi algorithm) in stereo matching. Baker first processed each pair of scanlines independently. After all the scanline matching was done, he used a cooperative process to detect and correct the matching results which violate the consistency constraints. Since this method does not use the across-scan-line constraints directly in the search, the result from the cooperative process is not guaranteed to be optimal. Ohta and Kanade applied dynamic programming for scanline search and across-scanline search. The problem of obtaining correspondence between edgels on right and left epipolar scanlines was solved as a path finding problem on a 2D plane. The problem of obtaining a correspondence between edges under across-scanline consistency constraint was viewed as a problem of finding a set of paths in a 3D space which is a stack of 2D planes for across-scanline search. The optimal path on the 2D plane is obtained by iterating the selection of an optimal path on the 2D plane is obtained by iterating the selection of an optimal path at each 2D node (intersection of left edge and right edge). Similarly, the optimal set of paths in 3D space was obtained by iterating the selection of an optimal set of paths at each 3D node. The numbers of computations involved are estimated to be O(T U2 V2 M2 N2) where T stands for the number of scanlines in the images, U and V are for the number of edges in the right and left images, and M and N for the average numbers of edgel points in one scanline in the right and left images, respectively. This is a tremendous number of computations involved in the search for correspondence.

In the Ohta and Kanade study the computation of cost in the search algorithm was based on the cost of the primitive path on a 2D search plane. The cost of a 2D primitive path was defined as the similarity between intervals delimited by edgels in the right and left images on the same scanline. The similarity was measured in terms of the variance of the intensity values of the pixels which comprise the two intervals.

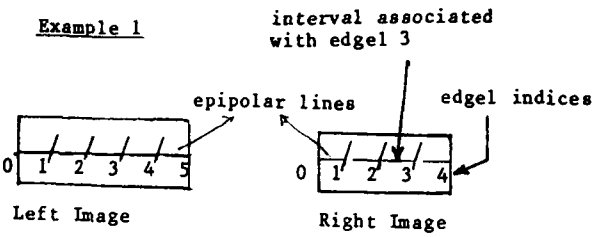
MODIFIED VITERBI ALGORITHM [8]

Dynamic programming is a technique useful for matching two sequences. It solves an N-stage decision process as N single-stage processes. This reduces the computational complexity to the logarithm of the original combinatorial one. The

viterbi algorithm is a dynamic programming algorithm which finds a best match from among all the allowable matches. The sequences to be matched define the two dimensions of a matrix such that each entry in the matrix represents the cost of matching that pair of elements. A path will consist of a sequence of nodes, each of which corresponds to one entry in the matrix. The goal is to find an optimal path through the cost matrix such that the sum of the costs along the path is a maximum. To determine the optimal path, two constraints are used: (1) The sequences must be matched monotonically and (2) each element of a sequence is used at most once. These constraints are equivalent to assuming that the path must start in the lower left corner and end in the upper right. From each node, a transition may only be made one unit vertically, horizontally, or diagonally [8].

In the case of edgel matching, the sequences to be matched are edgels from left scanline and from the corresponding right (epipolar line) scanline for 2D match. In the case of edge matching, the sequences are edges from left and right images. It is possible that certain sequence elements may have no match in the other sequence due to occlusion in the images. For unmatched elements zero cost is assigned. Vertical or horizontal transitions of one unit indicate occlusion of the element whose row or column is being entered. Diagonal transitions of one unit indicate a normal match associated with elements belonging to a newly entered row and column.

The viterbi algorithm proceeds by constructing a second matrix, of the same dimensions as the first, each entry with the accumulated cost of the optimum path from starting node to current node. The matrix values are filled in ascending order, left to right and bottom to top, beginning at the lower left. The transition rules guarantee that when it comes time to fill an entry its three predecessors will already have been assigned values. The algorithm simply examines horizontal and vertical predecessor accumulated cost values and the sum of diagonal predecessor accumulated cost value and the cost for the current position from the first matrix and then selects the maximum. This maximum becomes the current entry value for a second matrix and a pointer is stored to indicate which predecessor was selected. After the last position has been filled, the stored pointers are followed backward to the starting node, tracking out the optimum path from the upper right to the lower left. It is possible that the first or last few elements of a sequence are unmatched. This corresponds to allowing paths to begin at any point in the first row or column and end at any point in the last row or column. To achieve this, a zero row at the bottom and a zero column at the left are added to the first matrix. Then, the second matrix is constructed as before to obtain the optimal path which gives the best match. This procedure gives the same kind of results of Baker [16]. Fig. 2 illustrates the application of the modified viterbi algorithm for a simple example.



Left Edgel Indices	5	4	3	2	1
1	0	0	.1	.2	.9
2	0	.2	0	.7	.1
3	.1	0	.4	0	0
4	0	.2	0	.8	0

Right Edgel Indices

(First) Edgel-Cost Matrix

5	0	.9	1.6	2.0	2.8	occluded
4	0	.9	1.6	2.0	2.8	optimal
3	0	.9	1.6	2.0	2.0	path
2	0	.9	1.6	1.6	1.6	
1	0	.9	.9	.9	.9	
0	0	0	0	0	0	
	0	1	2	3	4	

Second Matrix Where Optimal Path and Matched Edgels Are Shown

Fig. 2 Edgel Matching

EDGEL CORRESPONDENCE SEARCH ON SCANLINES

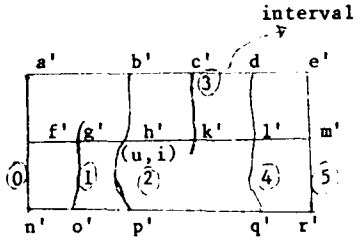
The correspondences between edgels (intervals) on the right and left epipolar (horizontal scanlines) lines can be solved on a 2D plane using the viterbi algorithm described in the earlier section. The edgels on each scanline are indexed in increasing order from left to right. Both ends of a scanline are treated as edgels. The cost of matching each pair of edgels (2D node) in the left and right scanlines are evaluated using a similarity function which is described in a later section. Using these edgel-costs as elements, an edgel-cost matrix (first matrix) is created. The viterbi algorithm is used to construct a second matrix which gives the optimal path and best edgel match. This is explained using Example 1. Unlike in references [8, 9, 10], in this study edgel correspondence search is used only to match edgels between edges on each scanline.

EDGE CORRESPONDENCE SEARCH ON THE TWO IMAGES

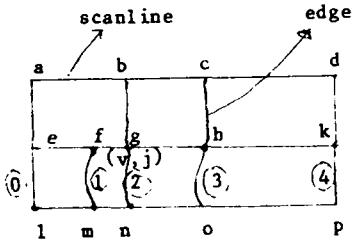
The edgel (interval) correspondences on scanlines alone cause ambiguity. To resolve this

ambiguity and to obtain global correspondences, edge (connected edgels across scanlines) correspondences are used. The problems of correspondences between edges (intervals between edges) in the two images can be viewed as an extension of scanline correspondence search.

For illustrative purposes, let us consider the right and left images in Fig. 3. The images are three scanlines wide. There are five intervals between edges (including right end; left end is treated as beginning) in the left image and four in the right image. (Even though the edges are shown as vertical lines, in reality, they are curves.)



Left Image



Right Image

The number in the circle next to an edge is its index.

Fig. 3 Images Showing Edges, Intervals and Scanlines.

A pair of edges in the left and right images make a set of 2D nodes when they share scanline pairs. This set of 2D nodes is referred to as a single 3D node. The cost at a 3D node is the sum of the costs of 2D nodes belonging to that 3D node.

For example, the cost of matching edge 2 in the right image with edge 1 in the left image (3D node 21) is the sum of the costs of matching interval fg with f'g' (2D node 21) on scanline 2 and mn with n'o' (2D node 21) on scanline 3. Similarly, the cost of matching edge 2 in the right image with edge 2 in the left image (3D node 22) is the sum of the costs of matching ab with a'b' on scanline 1, fg with g'h' on scanline 2, and mn with o'p' on scanline 3.

Using these edge-costs (3D node costs) as elements, an edge-cost matrix (first matrix) is computed. The viterbi algorithm is used to construct a second matrix which gives the optimal path and best edge match.

METHOD

In Fig. 3, edges belonging to left image and right image are shown. Even though the method matches edges in the two images to get depth information, in reality it is equivalent to matching surfaces between edges in the two images.

The method involves the following steps and computations:

(1) The edgels are obtained by applying the operator described in [7] to a pair of stereo images. The operator produces edgels by fitting a directional tanh-surface to windows in the images. These edgels are then linked.

(2) Edges are formed by an edge-linking process such that two edges in an image do not intersect.

(3) Edges are ordered and labeled in both images.

(4) The edgel (interval) similarity cost is computed from the formula:

$$\text{cost edgel}_{i,j}(t) = k \text{ (interval ratio similarity function)}$$

x (edgel orientation similarity function) (edgel contrast similarity function)

for all i, j, and t

where
i = index of edgel in the right image
j = index of edgel in the left image
t = scanline index
k = constant

All the above four steps are explained in detail in the implementation section later.

(5) The surface (interval between neighbor edges) similarity cost (edge cost) is given by

$$\text{cost}_{u,v}(s) = \sum_t \text{cost edgel}_{i,j}(t) = \sum_t C_{u,v}(t)$$

where u and v are edge indices in the right and left images, respectively. The summation is over the scanlines which are common to edge u in the right image and edge v in the left image. This forms one element (u,v) in the edge cost matrix which has dimension of U x V where U is the number of edges in the right image and V is the number of edges in the left image. The edgel j belongs to edge u and edgel j belongs to edge v.

(6) After determining the edge cost matrix (first matrix) from step 5, it is optimized using viterbi algorithm to obtain a second matrix as shown below:

$$cost_{o,u,v}(s) = Max\{(cost_{o,u-1,v-1}(s) + cost_{u,v}(s)), cost_{o,u,v-1}(s), cost_{o,u-1,v}(s)\}$$

$$cost_{o,o,o}(s) = cost_{o,u,o}(s) = cost_{o,o,v}(s) = 0$$

$$1 \leq u \leq U$$

$$1 \leq v \leq V$$

This gives the optimal path and best match of edges. The computations involved in this method for optimization are 3UV. Example 2 illustrates the steps involved in the method.

EXAMPLE 2

For Fig. 3, the edgel cost matrices are given for each of the scanlines. On scanline 1, there are 3 edgels (intervals) belonging to edges 2, 3 and 4, respectively, in the right image. Similarly, there are 4 edgels (intervals) on scanline 1 in the left image. Therefore the edgel cost matrix for scanline 1 is 3 x 4. For scanlines 2 and 3, the edgel cost matrices are respectively 4 x 5 and 4 x 4. These matrices are given in Fig. 4.

Left Edgel Indices

4	.1 25	.2 35	.4 45	
3	.3 24	.7 34	.1 44	
2	.4 23	.3 33	.2 43	
1	.8 22	.6 32	.3 42	
	1	2	3	

Scanline 1
Edgel Cost Matrix

5	.1 15	.1 25	.2 35	.3 45
4	.1 14	.2 24	.2 34	.2 44
3	.1 13	.1 23	.2 33	.3 43
2	.6 12	.7 22	.1 32	.2 42
1	.8 11	.3 21	.2 31	.1 41
	1	2	3	4

Right Edgel Indices

Scanline 2
Edgel Cost Matrix

4	.1 15	.1 25	.2 35	.6 45
3	.2 14	.1 24	.5 34	.3 44
2	.4 12	.1 22	.3 32	.2 42
1	.9 11	.3 21	.1 31	.1 41
	1	2	3	4

Scanline 3
Edgel Cost Matrix

Note: Double digit number in each element indicates indices of edges in the right and left images, i.e., 3D node numbers.

Fig. 4 Edgel Cost Matrices for the images in Fig. 3.

From the above three edgel cost matrices, each element in the edge cost matrix is computed by summing the costs belonging to the same 3D node. For example, 3D node 22 cost is computed by summing three values belonging to 3D node 22 on three scanline edgel cost matrices. The value is .8 + .7 + .1 = 1.6. The edgel cost (first) matrix and second matrix are given in Fig. 5.

Left Edges v

5	.2	.3	.6	1.3
4	.3	.6	1.4	.6
3	.1	.5	.5	.5
2	1.0	1.6	1.0	.7
1	1.7	1.6	1.0	.7
	1	2	3	4

Right Edges u

Edge Cost (First) Matrix

	5	0	1.7	3.3	4.7	6.0
Left	4	0	1.7	3.3	4.7	4.7
Edges	3	0	1.7	3.3	3.8	3.8
	2	0	1.7	3.3	3.3	3.0
	1	0	1.7	1.7	1.7	1.7
	0	0	0	0	0	0
		0	1	2	3	4

Right Edges

Second Matrix

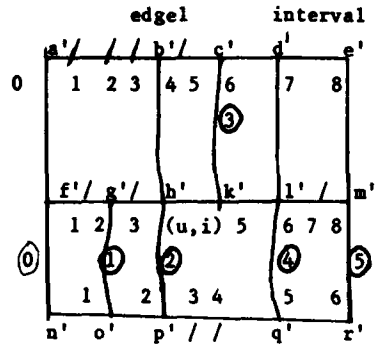
The path on the second matrix gives the best match. Matches are: (1,1), (2,2), (3,4), (4,5)

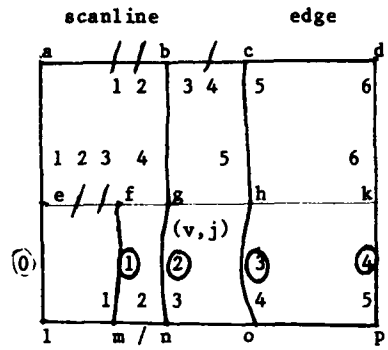
Fig. 5 Edge Costs and Optimal Edge Matches for the Images in Fig. 3.

MODIFICATION

In the above discussion we considered edges, which are longer than a threshold, running across several scanlines. If one desires to include edgels that do not belong to edges in the matching, the previous method needs modification. To study this, let us consider the right and left images shown in Fig. 6. This is similar to Fig. 3 except that edgels on individual scanlines are shown. The edgels lie between longer edges whose correspondence is determined by the method described previously. We want to determine the correspondence of these edgels (intervals) on right and left scanlines under the constraint of matching edges in the right and left images. The edge matching acts as a true (hard) constraint.

Two approaches are suggested below to modify the method described earlier to accommodate edgel matching between edges.





The number in the circle next to the edge is its index. On each scanline, edgels are labeled in increasing order from left to right.

Fig. 6 Images Showing Edgels, Edges, Intervals and Scanlines.

Approach 1:

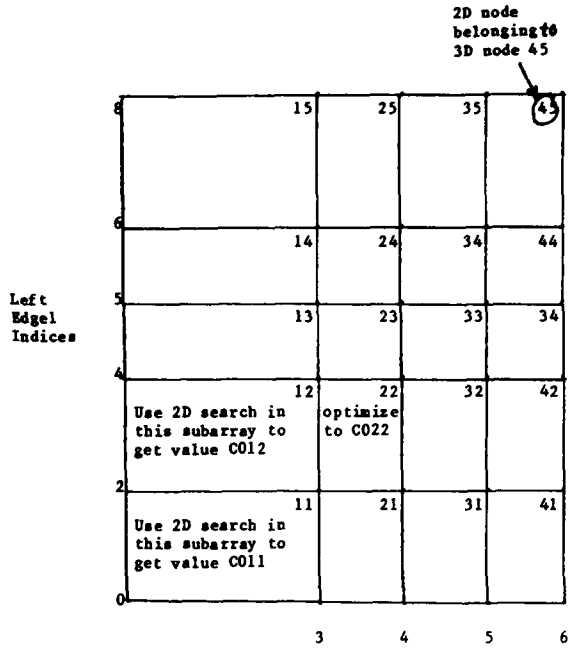
This approach involves modification of step 5 in the method. Suppose that edge 2 in the right image matched with edge 2 in the left image, then the edgels in ab need to match with edgels in a'b' on scanline 1, and edgels in fg need to match with edgels in g'h' on scanline 2 and edgels in mn need to match with edgels in o'p' on scanline 3. Therefore, we have multiple edgels (intervals), instead of a single edgel (interval), between edges on each scanline for matching.

For each scanline, we use 2D search to match edgels that are between edges. The optimal cost of matching edgels between edges is used to obtain edge cost as below:

$$cost_{u,v}(s) = \sum_{t} C_{o,u,v}(t)$$

The summation is over common scanline pairs (t is the scanline index).

The calculation of optimal cost for matching edgels between edges on scanline 2 is shown in Fig. 7.



For Scanline 2 in Example 2

Fig. 7 Illustration of the Calculation of the Optimal Cost of Matching Edgels between Edges.

The additional computation for this modification in the method is

T3MN

where

T = number of scanlines in the images

M = average number of edgels in one scanline in the right image

N = average number of edgels in one scanline in the left image

The total number of optimization computations = 3TMN + 3UV.

The estimate is obtained as follows: Each scanline requires 3MN computations, there are T scanlines, and edge matching requires 3UV computations.

Approach 2:

We first obtain edge matches using the six steps in the method and an appropriate threshold. Since the edge matches act as constraints, we can match the edgels in between matched edges by applying the 2D search only in the subarrays which belong to the matched 3D nodes (edges).

The total number of computations are much less than in Approach 1 because the 2D search is applied only in the subarrays belonging to matched 3D nodes ($\leq$ Min (U,V)).

IMPLEMENTATION

The stereo matching method described in this paper is implemented on a Symbolics 3600 machine using common LISP. The implementation details of the first four steps of the method are described below:

EDGEL DETECTION AND LINKING

We detect edgels by applying the operator described in [7]. An edge in an image corresponds to a discontinuity in the intensity surface of the underlying scene. It can be approximated by a piecewise, straight curve composed of edgels, i.e., short, linear edge-elements, each characterized by a direction and a position. The approach to edgel detection is to fit a series of one-dimensional surfaces to each window. The operator in [7] produces a list of edgels by fitting a directional tanh-surface to windows in the images. These edgels are then linked and fitted with conic sections and straight lines. The edgel detection is robust and has subpixel position accuracy and an angular localization better than 10 . Edges are formed by an edge-linking process such that two edges in an image do not intersect, and an edge does not cross a scanline more than once.

EDGE ORDERING

The edges are ordered by following a procedure which is similar to the one in [10]. The edges which run across the same scanline are locally ordered from left to right. This is done independently on each scanline. Then a graph representing this local order is generated with nodes as edges and directed arcs showing the local ordering between them. For each node, the maximum number of arcs from the leftmost node to that node is calculated. The edges are ordered by the increasing order of their maximum number of arcs. If the edges have the same maximum number of arcs, then the edge which starts earlier is given the smaller index. This assignment scheme is neither optimal nor unique.

EDGEL (INTERVAL) SIMILARITY COST

The step 4 in the method is the calculation of the edgel (interval) similarity cost. The similarity cost ($edgel_{i,j(t)}$) is based on the interval ratio similarity function, the edgel orientation similarity function, and the edgel contrast similarity function. We took similarity cost as proportional to the product of all three similarity functions. In computing the cost, we look for interval ratio similarity, orientation similarity, and the contrast similarity.

The interval ratio and edgel orientation measures have been used by Arnold [8] and Baker [9] in the cost evaluation function. They treated them as probabilities and evaluated the edgel cost.

In this paper, specific functions have been defined for these three measures and the following paragraphs explain these functions.

For an object surface, its image at a particular epipolar line will generally consist of two boundary edges and the interval between them. If the object surface is visible to both cameras, there will be a corresponding interval in each image. The lengths of these intervals are related to the angle of the surface and to the camera geometry. For small or moderate baselines, the lengths are comparable.

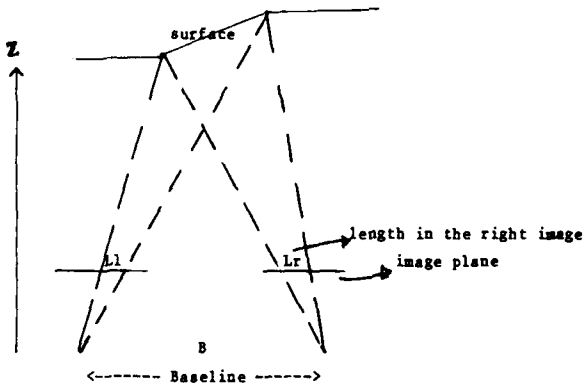


Fig. 8 Calculation of Interval Ratio Similarity Function.

Arnold [8] has shown that the probability density of the correspondence of the two intervals Ll and Lr in the two images is maximum when Ll = Lr.

We have chosen interval ratio similarity function fi to be

$$fi = e^{-10(1-r)^2} \quad \text{for } B/z = .07$$
$$= r \quad \text{for } B/z = .2$$

where

$$r = \frac{\text{Minimum (Ll,Lr)}}{\text{Maximum (Ll,Lr)}}$$

For a corresponding pair of edgels, one in the left image and one in the right image, we are interested in the similarity of their angles. We adopted the following similarity function, fo, for edgel orientation:

(1) if B/z = .07

$$fo = \left(1 - \frac{\Delta\theta}{3}\right) \quad \text{for } |\Delta\theta| < 3$$

$$= 0 \quad \text{for } |\Delta\theta| \geq 3$$

(2) if B/z = .2

$$fo = \left(1 - \frac{\Delta\theta}{15}\right) \quad \text{for } |\Delta\theta| < 15$$

$$= 0 \quad \text{for } |\Delta\theta| \geq 15$$

where

$\Delta\theta$ = $(\theta_l - \theta_r)$

θ_l = edgel orientation with respect to epipolar line in the left image

θ_r = edgel orientation with respect to epipolar line in the right image

$\theta_r > 10$

and

$\theta_l > 10$.

The edgel contrast similarity function, fc, is given by

fc = 1.0 if contrast of edgel in the left image = contrast of edgel in the right image

= 0 otherwise

RESULTS

This paper addresses mainly the identification of corresponding points in the two images. Since our method generates only a sparse disparity map, it is difficult to display the results in a very convenient visual form. Proper display requires the construction of an interpolating surface [17], and good interpolation requires object segmentation first. This is not done here because it is a large problem by itself.

The matched segments in the left and the right images are displayed separately. The information about which segments (edges) in one image match with which in the other image is provided.

STANFORD UNIVERSITY BLOCKS IMAGE

We first applied our method to this image of some blocks, cylinders and spheres as shown Fig. 9. The image size is 256 x 256 pixels and the intensity resolution is 8 bits. Fig. 10 is the result from edge detection, linking and thresholding (on the length of edges: pixels). The number of edgels in the right and left images is about 1700. The edges in each image is about 55. The number attached to each edge indicates its ordering index.

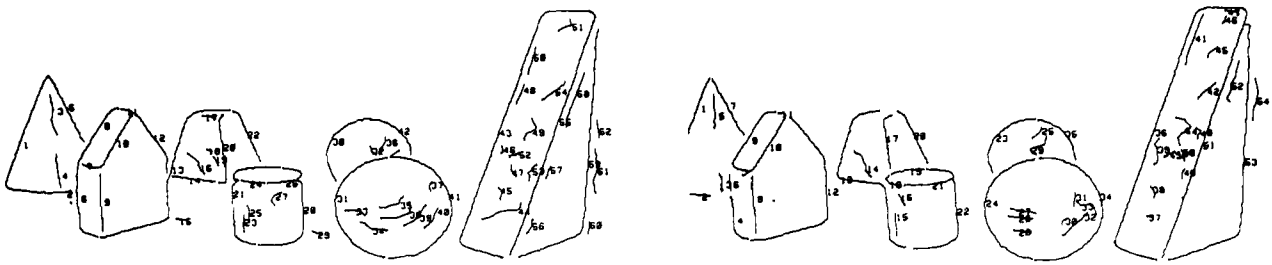
Fig. 11 indicates the matched segments (edges). The number attached to each edge in this figure still indicates its ordering index. There were few edges in Fig. 11 which were not matched even though the visual inspection indicates that they should have. This can be mainly attributed to the nonunique ordering of edges in the two images. The omitted edges which have the same number of maximum number of arcs are given an arbitrary ordering index following a procedure suggested in [10]. No incorrect matches were observed. We essentially captured the shape of the objects in both images except for two edges in the left part of the images which were not matched due to the arbitrary indexing of these edges.

AERIAL VIEW OF WHITE HOUSE

The second image we took is a 256 x 256 portion of the 512 x 512, 8-bit intensity resolution picture of the White House (Fig. 12). This is an interesting image because it contains buildings and highly textured trees. Fig. 13 is the result from edge detection, linking and thresholding (on the length of edges). The number of edgels in the left and right image is about 3800. There are about 140 edges in each image. As in the previous example, the number attached represents

Left Image Right Image

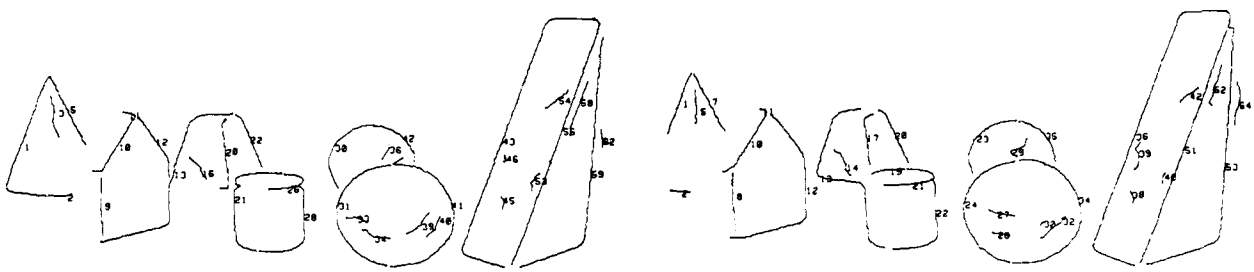
Fig. 9 Stanford University Blocks Image



Left Image

Right Image

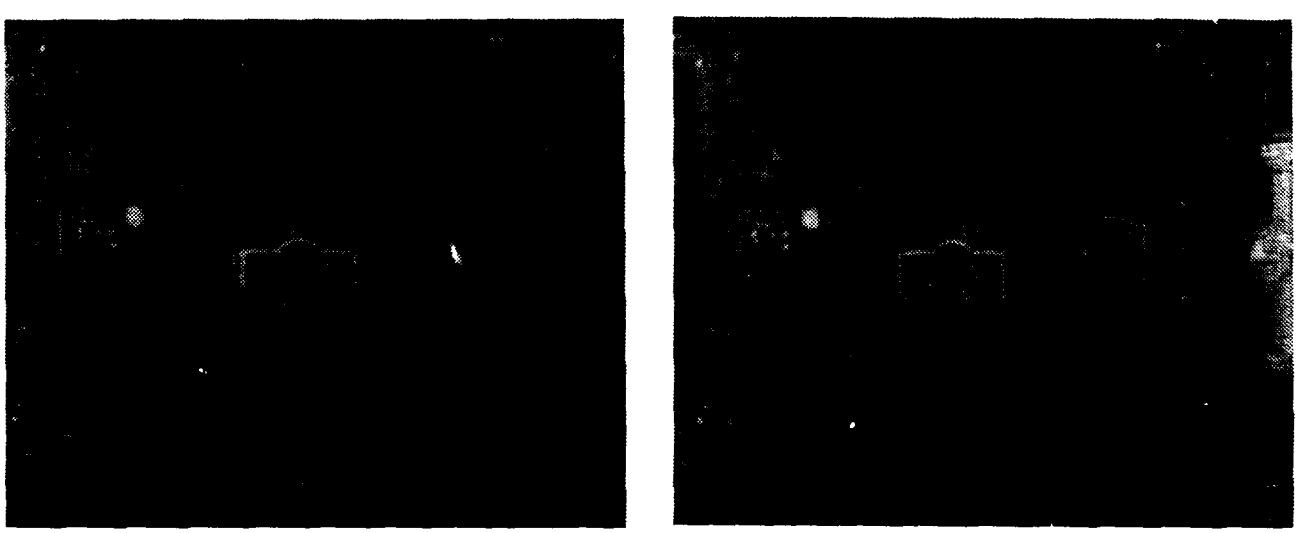
Fig. 10 Edges in the Right and Left Images



Left Image

Right Image

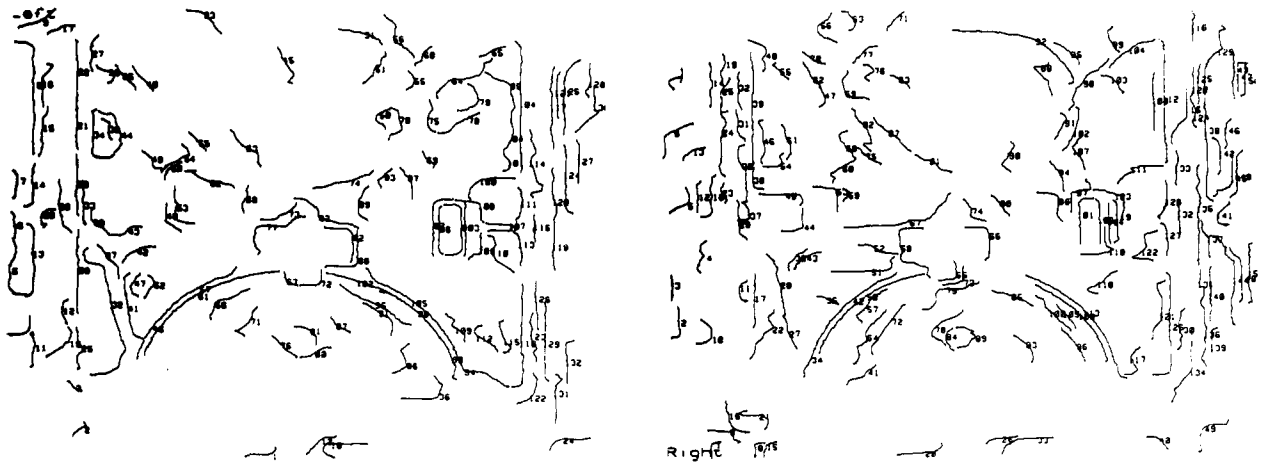
Fig. 11 Matched Edges (Segments)



Left Image

Right Image

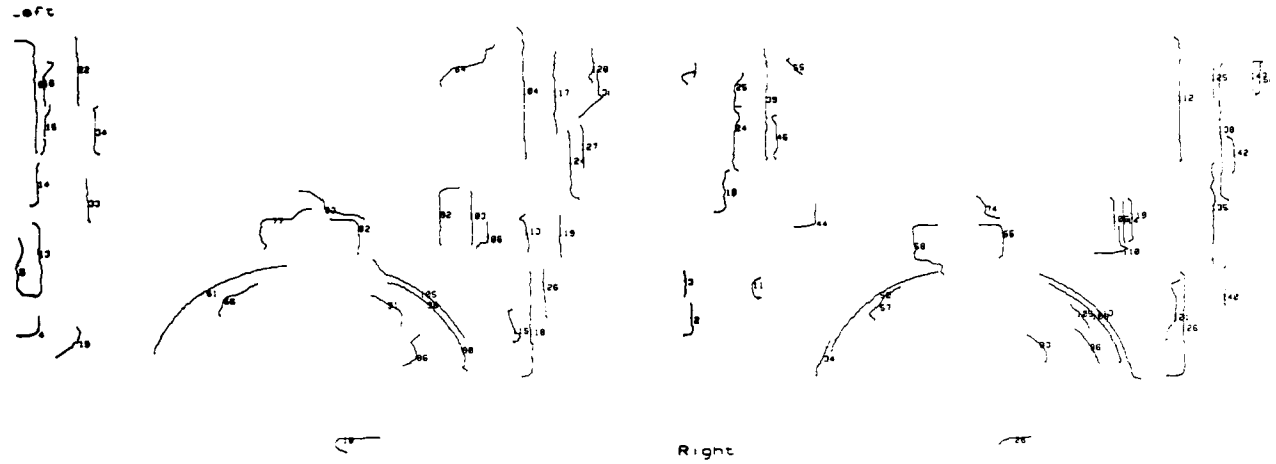
Fig. 12 Aerial View of White House



Left Image

Right Image

Fig. 13 Edges



Left Image

Right Image

Fig. 14 Matched Edges

the ordering index. Fig. 14 indicates the matched segments (edges). Although basic feature elements have been successfully matched, the percentage of matched segments is only 25% of total segments. This small percentage is due to an arbitrary ordering scheme. We consider the current results a preliminary finding.

CONCLUSION

We have described a stereo matching method which provides optimum matching for edges (intervals between edges) in the two stereo images. The algorithm requires only 3UV computations for edge matching and additional 3TMN computations for matching edgels in between edges. The method works well for images which have long edges. The method offers an efficient way for stereo matching.

ACKNOWLEDGMENTS

The authors would like to extend their gratitude to V.S. Nalwa for providing the necessary data for the work. We thank David Kriegman for his help in the display programming and Ross D. Shachter for useful discussions.

REFERENCES

1. D. Gennery, Object detection and measurement using stereo vision, in Proc. Image Understanding Workshop, College Park, Md., Apr. 1980, 161-167.

2. H. Moravec, "Obstacle avoidance and navigation in the real world by a seeing robot rover," Stanford Artificial Intelligence Laboratory AIM 340; Ph.D. thesis, Stanford University, Sept. 1980.

3. R. Henderson, R. Miller, and C. Grosch, Automatic stereo reconstruction of man-made targets: Digital processing of aerial images, Proc. Soc. Photo-Opt. Instrum. Engr. 186, No. 6, 1979, 240-248.

4. R. Kelly, P. McConnell, and S. Mildenberger, The gestalt photomapping system, Photogrammetric Eng. Remote Sens. 43, No. 1407, 1977.

5. D. Panton, A flexible approach to digital stereo mapping, Photogramm. Eng. Remote Sens. 44, No. 12, 1978, 1499-1512.

6. S. Barnard and M. Fishler, Computational stereo, ACM Computing Surveys 14, No. 4, 1982, 553-572.

7. V.S. Nalwa and T.O. Binford, On detecting edges, IEEE-Transactions Pattern Anal. Mach. Intell. PAMI-8, Nov. 1986, 699-714.

8. R.D. Arnold, "Automated stereo perception," Stanford Artificial Intelligence Laboratory Tech. report AIM-347, Stanford University, 1982.

9. H.H. Baker and T.O. Binford, Depth from edge and intensity based stereo, in Proc. 7th Int. Jt. Conf., A.I., Aug. 1981.

10. Y. Ohta and T. Kanade, Stereo by Intra- and Inter- scanline search using dynamic programming, IEEE Trans., PAMI, Mar. 1985.

11. M.J. Hannah, "Computer matching of areas in stereo imagery," Ph.D. thesis, Stanford University, 1974.

12. D.J. Panton, A flexible approach to digital stereo mapping, Photogramm. Eng. Remote Sens. 44, No. 12, 1978.

13. G.M. Medioni and R. Nevatia, Segment-based stereo matching, Computer Vision, Graphics, and Image Processing 31, 1985.

14. D. Marr and T. Poggio, "A theory of human stereo vision," MIT AI memo, No. 451, Nov. 1977.

15. D. Arnold and T. Binford, Geometric constraints in stereo vision: Image processing for missile guidance, Proc. Soc. Photo-Opt. Instrum. Engr. 238, 1980, 281-292.

16. H.H. Baker, "Depth from edge and intensity based stereo," Stanford Artificial Intelligence Laboratory Tech. report AIM-347, Stanford University, 1982.

17. W. Grimson, From Images to Surfaces, MIT Press, Cambridge, Mass., 1981.

18. H.S. Lim and T.O. Binford, Stereo correspondence: Features and constraints, in Proc. Image Understanding Workshop, Dec. 1985, 373-380.

Steps Toward Accurate Stereo Correspondence[1]

Steven D. Cochran

Institute for Robotics and Intelligent Systems
School of Engineering
University of Southern California
Los Angeles, California 90089-0273

Abstract

This is a preliminary report for a system which obtains an accurate determination of 3-D surfaces from passive stereo. We present an overview of the proposed system, which encompasses aspects of *Image Matching*, *Depth Determination*, and *Interpolation*. In *Image Matching* a major source of error in the exact matching of edge elements is the loss of edgels near corners. A new method is presented which will fill these gaps in a reasonable manner. The approach used first marks the junctions in the image with a labeling which includes the orientation of the associated edge gradients. Next, those edges which lie near the boundary of blobs in the image are checked and where problem junctions occur (violations of an "*at least one-in and one-out*" rule), a search is made along the boundary of the associated blob for the best candidate to complete the boundary. Initial results for the gap-filling process are shown for real images.

1 Introduction

Human Beings are able to perceive depth in 2-D "monocular" images and in an enhanced form from the stereoscopic combination of a pair of images. The process used by the Human Visual System to do this, however, is not well understood. Much research has been devoted to the automatic abstraction of information about objects in images, in order to produce autonomous systems which are able to "perceive," and operate upon their environments and, in some cases, to gain insight about the operation of the Human Visual System.

The major problem with current stereo correspondence algorithms such as [1,2,3,4,5,6,7,8,9], is that they have problems when dealing with added or missing features due to noise or occlusion in real images. In addition, it is often difficult to determine in detail the correct disparity of the matched features in feature-based methods since the same abstraction that allows a feature to be represented also introduces some ambiguity in how to provide a point-by-point pairing of the feature in one image with its corresponding "fine-structure" of the feature in a second image.

The recovery of the 3-D characteristics of a scene from multiple images taken from different points of view, may be viewed as consisting of the following steps (from Barnard and Fischler [10]):

1. Image Acquisition
2. Camera Modeling
3. Feature Acquisition
4. Image Matching
5. Depth Determination
6. Interpolation

We plan to solve the *Depth Determination* step of this process. In order to do this we will concentrate on the last three steps, detecting and correcting errors in the *Image Matching* step, performing a partial *Depth Determination*, followed by an estimation of the object surfaces which will allow the *Interpolation* step to re-analyze the matching to give a final, accurate, determination of the depth.

1.1 Initial Results in Gap-Filling

We present, in this paper, our initial results in gap-filling as an initial step in the correction of the *Image Matching*. In our analysis of the gap-filling, we concentrate on detecting the gaps formed at corners since they are a particularly interesting example and the technique simplifies to handle more general edges.

Corner detection is an important research area in computer vision and image processing. When complete curves are present, the task is the determination of which point or points form the corner and finding its radius of curvature. Various corner detectors of this sort have been developed and are reported in [11,12,13].

[1]This research was supported by the Defense Advanced Research Projects Agency under contract F33615-84-K-1404, monitored by the Air Force Wright Aeronautical Laboratories, DARPA order number 33119, and by a TRW Fellowship.

A problem occurs when the curve is fragmented due either to the assumptions made by the underlying edge detection or to a low edge contrast. The former may be improved by better edge detection algorithms, but the latter remains as a source of gaps in the curves which are often accentuated at the corners due to the interaction of two (or more) edges. Processing the intensity images in Figure 1 yields several examples of such lost corners, these are shown in Figure 2. When a complete curve is not available, some method must be introduced to fill-in the missing points. One example of this approach is [14], in which Nevatia and Huertas use knowledge of the scene domain to predict the expected corner and then use shadows to confirm their hypothesis. Their hypotheses assume approximately orthogonal **L**- or **T**-junctions. While this is useful for some domains like building detection in aerial images, it does not take into account non-parallelepiped structures such as those in Figure 1 (a). We propose to *combine* the information extracted from the original intensity images via both feature- and area-based approaches.

2 System Description

We have decided to use a feature-based approach since our desire is for an **accurate** system. The feature-based methods give us the potential for sub-pixel accuracy at the corresponding points which in turn gives us a better triangulated range. The major problems with these methods are:

1. Missing and Extra features,
2. Mismatched features,
3. Unmatched features, and
4. Sparsity of data.

Accuracy in a two camera system is best along edges perpendicular to a plane through the baseline between the cameras. This is because the epipolar geometry provides an additional source of information which is lacking when the corresponding points are **contained** in such an epipolar plane. Thus not all point correspondences are equally good.

We plan to augment a feature-based method developed by Medioni and Nevatia [4] to detect and correct all of these problems. Mismatches will be identified and corrected by extending the work of Mohan [6] from edgels to contours. Missing edges and gaps will be filled by a modified form of model matching after Falk [15], Shirai [16, 17] and Shapira and Freeman [18], along with a newly developed join labeling scheme and the gap-filling (both described in this paper). Surfaces will be located by analysis of contour and surface marking edges to define a dense matching.

We present below, an overview of our proposed system for the detection and correction of stereo-correspondence errors that:

1. Integrates two- and three-dimensional continuity.
2. Hypothesizes corrections to the observed image features.
3. Performs *exact* matching of the features at the pixel (or subpixel) level to determine disparity where possible and marks those places where disparity *cannot* be determined without inferring the underlying geometry of the object or objects making up the scene.
4. Hypothesizes faces or surfaces in three dimensions consistent with the observed and matched features.
5. Propagates disparity across those features whose disparity could not be found in the third step above, guided by the hypothesized surfaces.

We use the hypothesize-and-test paradigm in three different areas to detect and correct errors in matching, to locate missing features, and to build object surfaces. When possible, each hypothesis is verified and the matching is corrected by using the connectivity and geometry of the 3-D scene being constructed. High confidence matches are initially accepted as valid hypotheses and disparities assigned. Where a hypothesis cannot be verified, it is carried forward in the analysis. In the event that a hypothesis *cannot* be verified within the scope of our analysis (for instance, some perceived edges may be missed) it becomes part of the analysis and serves the purpose of alerting *higher-level* processes of an unresolved problem.

We assert that typical errors in passive feature-based stereo-correspondence, such as those listed at the beginning of this section, may be corrected by the reprocessing of the matched features in two or more images. Also, that in order to assign disparity, the matching between features *must* be extended to its component parts which *cannot* be completely accomplished without first inferring the object surface that generated these features.

The initial stereo matching allows us to investigate the monocular continuity of lines in one image relative to the other. Inconsistencies indicate errors in either the feature extraction or in the matching. Such errors are usually caused by noise or occlusion: but sometimes are due to a missing line whose existence must be hypothesized from the image data and knowledge about the world. (These later sources of error are called *perceived edges*.) During this phase we seek to directly correct as many errors as possible and attempt to detect the rest, although some errors may not be detectable at this stage. The detected inconsistences which cannot be corrected generate hypotheses which are used as markers to guide the application of information gathered during the subsequent processing.

The ultimate goal of this system will be to provide an *accurate* and *robust* stereo matcher that correctly matches a set of features in an image pair. In addition we will be able to provide a dense disparity map for the scene which is

consistent with the observed features and that is produced using passive imaging techniques. A list of unresolved hypothesis/problem entries will serve to alert higher-level processes of problems with the interpretation which may need the application of more scene specific knowledge such as assumptions of object type to be resolved. In addition, feature abstraction, labeling and matching information will be available so that it will be possible to feed back information to the system or for it to serve as additional input to the high-level processing. We view the proposed system as being an important link between existing low and medium level feature extraction systems and higher level systems such as ACRONYM [19] or as an additional channel for systems like MOSAIC [20].

2.1 Choice of Primitives

As part of our overall strategy we will be using the following features to extract accurate stereo correspondences between points in a pair of passively acquired intensity images. The purpose of this paper is to show how the missing corners may be found. We are especially interested in junctions into which there are *no* edges entering or from which there are *no* edges exiting. The most common such junction is the **E**-junction which represents the endpoint of an isolated curve or segment.

The problem that we wish to solve is that of finding an **accurate** representation for the visible surfaces of a set of objects in a scene, by the analysis of a pair of stereo intensity images of that scene taken by two cameras a small distance apart. Therefore we use feature-based methods, since they provide better accuracy by matching features which may be found to sub-pixel precision. The feature primitives that we will use are *edgels*, *segments*, *junctions*, *regions*, and *blobs*. We choose these features because they are indicative of intensity discontinuities which, typically, occur only at object boundaries, at surface discontinuities and across surface markings (eg. in textured regions).

EDGELS are the fundamental feature that we use. They represent the location and orientation of a zero-crossing of the second derivative of the intensity image, located to pixel or sub-pixel accuracy. Their orientation is perpendicular to the direction of the maximum change in intensity such that the lighter side is to the right. They will serve as the *source* of our most accurate information about the position of interesting points. We divide the edgels by two classifications into four groups: the stronger and the weaker, based on (1) their strongest response to an edge mask and (2) their alignment with respect to nearby strong edgels; and vertical —vs— horizontal, based on their alignment with respect to the epipolar rays.

CURVES represent connected sequences of edgels. They provide an ordered progression through the image and may be used to propagate (both monocular and stereo) relationships through the image, subject to the restriction that they may (1) have some non-invariant shifts, *e.g.* specula-edges shift with even small changes in viewpoint; (2) wander from one object to another which is adjacent in 2-D but not necessarily in 3-D.

LINEAR SEGMENTS are a further abstraction of the edgels and represent straight or nearly straight sequences of them. These are the most useful features for stereo matching because they are simple and closely match the important edges in the scene. However, their accuracy is *not* very good and they should not be used for depth determination. Segments may be labeled as being on the boundary or in the interior of a blob (see below).

JUNCTIONS (or JOINS) are the edgels at the intersection of curves and/or linear segments. They are used to indicate points at which matching may be propagated due to continuity. In addition, some join types are very unlikely. These unlikely joins will be used to suggest possible problem areas such as those places where segments were missed. We use a labeling scheme in which we retain a list of properties at each junction. These properties include:

- The total number of intensity edges entering or leaving the junction (the minimum is 1, the maximum used is 3).
- The number of intensity edges entering the junction, and hence (along with the total edges count) the number leaving.
- The junction type — in the traditional **L**, **T**, **Y**, **W** sense, to which we add **S** for nearly straight sequences, **E** for lone endpoints, and **X** for junctions composed of more than three edges.
- The arrangement of the junction in terms of the general angle between the edges. We use the values:

  A Acute

  L Square (approximately)

  O Obtuse

  S Straight (approximately)

REGIONS are another way of viewing the intensity discontinuities. We will use them to help with gap filling and with grouping together features which form the members of object faces. We locate regions by recursively splitting an image using a threshold defined by histogram after [21]. Associated with the regions are simple attributes such as *area*, *perimeter*, average-intensity, and *orientation* as well as higher-order attributes such as $perimeter^2/area$ and the *s.d.* of *intensity*.

BLOBS are a subset of the regions which meet the following set of criteria. These criteria are designed to admit only the semantically meaningful regions while excluding the rest.

1. The regions must have no descendant. That is, it must be a leaf-node in the hierarchical tree of regions.

2. Its area must be greater than some minimum.

3. The region must be compact.

However, we desire to accept and work with real world scenes and therefore we do **not** assume that our data is perfect, rather, we expect missing and extra edges and junctions due to noise and conditions that we *assume* do not exist in our analysis but which do *occasionally* occur and produce errors during the low-level and preprocessing steps. We feel that a realistic system must be prepared to detect and correct such errors while at the same time choosing heuristics which generally serve to quickly and accurately determine the components of the scene.

During the 2-D processing we use **both** of a pair of stereo images: where at first, one is considered to be our monocular image while the second acts as a model, and then they swap roles. An initial mapping between the image and the model is provided by a stereo matcher [4] that provides a list of likely matches between the segments in the two images.

2.1.1 Join Labeling

The following are the rules that our system uses to label joins. The *type* of join is determined by the *number* and *orientation* of the lines which form it, while the subscript in our nomenclature gives the number of those (intensity) oriented segments *entering* the join. The superscript indicates the specific join from the variations possible under our "ALOS" model. Each such variation is unique in terms of rotation and reflection from any other variation. The variations show the combinations of acute (A), right (L), obtuse (O) and straight (S) angles combined with the relative line orientations. This alternate labeling scheme will be used to provide an additional measure of similarity between junctions, in which the angles between the segments changes between the two viewpoints. This will be described in more detail in Section 2.2.2. The angles are termed *right* if they are within **L-Threshold** of 90°, and *straight* if they are within **S-Threshold** of 180°.

1. 'E'-junctions represent the endpoints of segments that do not form a join.

2. Joins of exactly 2 segments are labeled as one of the following two types:

   (a) **L**-junctions are those joins between exactly two segments which form an angle which is not considered **straight**.

   (b) The remaining joins are considered present only for the linearization of curves to simplify analysis (e.g. knot points). For labeling purposes, the joins are marked as **S**-junctions indicating that they are, in some sense, straighter than **L**-junctions.

3. Joins of exactly 3 segments are labeled as one of the following types:

   (a) **T**-junctions if one pair of segments form a **straight** angle.

   (b) **Y**-junction if all clockwise angles between pairs are less than 180°− **S-Threshold**.

   (c) **W**-junction if one clockwise angle between pairs is greater than 180°− **S-Threshold**.

4. Joins of higher order (i.e., composed of more segments) than 3 are labeled as *multi*- or **X**-junctions and will be further expanded only if junctions of this type occur often enough to make this extension worthwhile.

Junction Catalog Figure 3 shows all possible first, second and third order joins. Some of these are very unlikely because they require a smooth, low spatial frequency, change in intensity locally about the join, in violation of the generality assumption, those which are considered likely are labeled as "complete." The unlikely cases can be used to support the hypothesis of, or a search for, a more likely (complete) join type. For instance, if a join of type L_2^a were found then it would support the hypothesis that one of Y_2^{abc}, W_2^{abcde}, or perhaps one of T_2^{ac} may actually be present. We submit that a corollary of the *generality constraint* is that for a join formed by the meeting of lines corresponding to intensity gradients in the image, at least one segment must enter the join and one segment must exit the join.

2.2 2-D Processing

During the 2-D processing phase of our system we integrate and organize the information supplied by the preprocessing routines above. Segment match information, along with the continuity between adjacent segments and the proximity of other features, are used to detect and correct some of the matching errors produced during the preprocessing. In some cases a problem may be detected but the correction cannot be determined. We retain this information by adding one or more problem/hypothesis entries to the list of unresolved hypotheses.

2.2.1 Continuity Analysis

The set of connected joins may be used to propagate matches (indicated by a "⇔") through the image. If we

assume that the disparity changes smoothly over each line, then if there are a few incorrectly matched points along the line, they may be corrected by setting them to values interpolated along the line [6]. This process may then be extended by allowing the values to propagate along connected (indicated by a "$\rightsquigarrow$") lines *whether or not they have been matched*, except that no information may be propagated through the "stem" portion of the 'T'-junctions. Also, the *goodness* of the match varies with the orientation of the lines relative to the local epipolar-plane(s).

For instance, if we have two matched curves $C_l \Leftrightarrow C_r$, as shown in Figure 4, we may hypothesize that $A_l \Leftrightarrow A_r$, $B_l \Leftrightarrow B_r$, $D_l \Leftrightarrow D_r$, and $E_l \Leftrightarrow E_r$.

Incorrect matches may also be detected by continuity. Figure 5 shows a case of a bad match $B_l \Leftrightarrow C_r$, where $A_l \Leftrightarrow A_r$ are matched and B_r is unmatched. Here, the connectivity of $A_l \rightsquigarrow B_l$ and $A_r \rightsquigarrow B_r$ along with the matching above implies that the $B_l \Leftrightarrow C_r$ match should be broken and replaced by a hypothesized $B_l \Leftrightarrow B_r$ match. In the case where connectivity alone does not allow a choice of which matches to prefer, then the "goodness" of the match may be used to resolve the tie.

2.2.2 Hypothesis Formation

The rules for Hypotheses formation based on join labeling are:

1. There is always at least one line entering and exiting a junction. This comes from the assumption that no chance alignment occurs, and allows us to hypothesize in many cases the existence and approximate orientation of additional edges that were missed during the feature extraction. Often these hypotheses will later serve as the guides for the Gap-Filling process.

2. Matched junctions have the same number of incoming and outgoing edges. This rule is weaker, but we are assuming acute-angle stereo which the baseline is much shorter than the range to the scene. In this scenario, the vertices generate similar junctions in the image planes and with the small changes that can occur. Figure 6 shows a graph of transitions between junction types.

   This allows us to hypothesize the most likely matching junction as well as less acceptable alternatives should other evidence support an alternative match. These hypotheses when combined with the continuity and the partial match data will allow us to correct most of the errors which occur during the matching phase. Table 1 gives the distance between the junction types shown in Figure 6, assuming a unit cost for each transition. A cost of 0–2 is reasonable, but higher costs imply an unlikely match.

For each *incomplete* junction, the component lines are extended along their orientation, and along any associated region, for a set length and a search is made within an associated sweep angle. Both of these parameters are calculated locally based on the length and orientation of the individual segments and the strength of the region boundaries.

2.3 Gap Filling

From the passively acquired intensity images we extract the edge points and organize them into linear segments. At the same time we extract the hierarchical tree of regions.

In the actual system the linear segments in a pair of stereo images are matched and the match data is checked by using the continuity in one or both images. When corners are lost, it is often the case that the corner is lost in both of the images, so we will consider only the monocular application at this time, since continuity will serve to fill the gap when only one of the stereo images contains a gap.

For each likely endpoint we will look for the nearest blob boundary point and follow the contour of that region until we locate an acceptable continuation segment/curve. Then we must determine the exact path of the corner.

2.3.1 Initial Processing

We first apply a feature- and an area-based operator to the intensity image. The feature-based operator extracts the edges from the image, thins and links them using the technique developed by Nevatia and Babu [22]. This process yields both the edgels linked together to form curves, and the linear segments abstracted from these curves. The second operator extracts a hierarchical tree of regions using the Ohlander-Price-Reddy algorithm [21]. Since we are interested in regions that are as compact and convex as possible, we further process the regions filtering out any regions with an area smaller than 100 or with a value of $perimeter^2/area > 30$ to obtain the desired regions, which we now term "blobs."

2.3.2 Associating Segments with Blobs

Next each segment is potentially associated with a blob by searching a square window of half-width w about each of the segments component edgels (see Figure 7.

A count of the edgels within the search area is maintained by blob-id and whether the edgel is on the boundary or in the interior of the blob. The points on the boundary are weighted with a value of $2.5w$, which is the relative importance of a boundary point to an interior point in the window. The total weighted points are summed and the segment is assigned a classification by the percentage of the points in each category. The following is a typical classification which indicates that it is very likely that the point is along the boundary of blob 14 but should also be considered when working with blob 11.

```
(:CLASSIFICATION
     (:BOUNDARY 14 0.46)
     (:BOUNDARY 11 0.29)
     (:INTERIOR 14 0.15)
     (:INTERIOR 11 0.10))
```

2.3.3 Locating the Endpoints of a Gap

A search is made near the endpoint of a selected segment and a boundary point is selected. The boundary is followed in the direction indicated by the type of junction and the gradient across the region boundary. An endpoint of type $\mathbf{E}_1$ (where the subscript indicates the number of incoming edges) indicates that the search should check for an outgoing edge while following the boundary. For instance, for a region with a light interior the search would be clockwise around the region. Figure 8 shows the search region along the boundary of some example regions.

As with the classification of segments above, we use a window of half-width w and search for a compatible segment in the border-set of the region. For the $\mathbf{E}_1$ junction example above, we would desire an $\mathbf{E}_0$, **S** (to form a **T**-junction), or any other junction which needs or will accept an incoming intensity edge.

2.3.4 Determination of the Corner

Now that the endpoints of the missing corner have been located, as well as the associated section of the blob boundary. We must determine the specific path of the corner. For monocular images this may be done by:

1. Extending the chosen curves until they intersect, for a pair of curves that are approximately orthogonal, or
2. Following the smoothed contour of the blob at the distance $(1-t)\,d_1 + t\,d_2$ from the boundary, where t varies from 0 to 1 along the gap, and d_i is the distance of the endpoints from the blob.

For stereo images, there is an additional constraint of desiring a smooth transition on the assumption that the gap being filled represents a continuous contour of an object face [23,6].

2.3.5 Results

Figure 9 shows some of the strong segments in each of the images. The segments marked as being very likely on the boarder of the center blob are shown with darker edgels (the circles mark the direction of the edge in the manner of an arrowhead). Figure 10 shows the application of this process to a gap in each of these three different domains. Note that in Figure 10 (a) the search rejected a possible segment because it could not form a reasonable junction, and in (c) the vertical segment boarding the nearby region was rejected in preference of a better candidate above, because the nearby segment was more likely part of another blob.

In combination with continuity checking [24,3], filling the gaps by combining information obtained from multiple sources can help to improve the matching of contour edges. This, in turn will allows the detection and correction of errors in stereo matching. The improvement in exact edgel matching is an important part of accurately determining the depth of points in 3-D.

2.4 3-D Processing

During this phase of the processing we plan to use the feature primitives along with their 2-D matching and the camera model to generate a set of points in 3-D. The first sub-task is to generate the detailed matching (at the pixel or sub-pixel level) of the matched features. As discussed below, we will not be able to generate all of the matches initially. Next we will cluster the points representing features that seem to be associated with individual object faces and once surfaces have been determined, they will be used to find the remaining edgel matches. In addition, refinements will be made at this to correct for distortion caused by vertically oriented limb edges.

2.4.1 Exact Edgel Matching

Horizontal disparities carry information about the local depth variations, while vertical disparities (because they are *insensitive* to depth) can be used to extract the viewing system parameters of angle of gaze and viewing distance without requiring stronger assumptions about the nature of the system from which they originate.

Currently edges are extracted at approximately pixel precision, that is, each edge point is accurately positioned to within $\frac{1}{2}$-pixel of its actual location. The line segments that represent approximately linear runs of edgels vary from the assigned edgel locations by at most a specified maximum. We value long reasonably straight runs of edgels as a useful feature for stereo matching but the errors associated with these features are to great for the derivation of *accurate* 3-D data. To get the accuracy that we need we first note that when we match segments we are only using the segments as a useful approximation to a linear run of edgels so it is the edgels that we want to match. However, the edgels have an error also. Research is being done at USC by Huertas, Medioni, Chen, and Ulupinar [25,26] to obtain edgels to sub-pixel precision and we plan to use this technology as it becomes available. In the meantime we will simulate this by fitting a B-spline curve locally to the edgels to guarantee that we have a C_1 smooth curve in 3-D when we match the curves.

However we must also consider those cases where we cannot assign a disparity at all because the edge is *horizontal* with respect to the epipolar plane. Such edges cannot be assigned a disparity because stereo correspondence is based

on the assumption that we have a single point correspondence in the epipolar-plane. But in the special case where the edge is in the epipolar-plane we will have many such points. There are only two ways that we can assign disparity. The first is that we *assume* a linear fit and we match points in the ratio of their lengths. For some scene domains [2,27], we may do this but there still remains the problem of missing edges or extra ones due to noise. If we do not know the endpoints then we may not match the points at all.

In scenes which contain curved objects, this method will not work because the matching relationship between the points on the two segments is nonlinear. In fact when a curve is terminated by a limb (self occluding) contours, such as the top or bottom of an upright cylinder, we do not know the correct end points. For a method to work with all such scenes then, we need a more general approach.

2.4.2 Grouping Segments in 3-D

The one that we propose to use is to hypothesize a surface which:

1. Matches the *vertical* segments.
2. Does not disagree with other hypothesized surfaces.
3. As much as possible, with non-perfect data, incorporates the observations of Lowe and Binford [23].

This surface may then be used to propagate the known disparities at *reliable* contours along the *horizontal* contours, and along areas where the feature extraction and/or the matching was incomplete. In addition this surface serves as a hypothesis of a dense disparity map which is consistent with the observed images.

2.4.3 Edge Analysis and Vertex Detection

Once we have found the surfaces making up the faces of the objects in the scene, we want to use those surfaces to verify the the edgel matching which was given a low weight (such as those along *horizontal* edges). Also, during this sub-task, we label the edges and vertices. To do this the intersections of adjacent object faces are calculated. Some adjustments to the surfaces may be necessary to preserve the location of high-confidence points such as well defined vertices. Once all of the surfaces have been found, the wire frame is re-analyzed to determine the edgel matching along *horizontal* edges in the scene, which could not be given a good match disparity prior to the surface generation. The contours and vertices of the objects in the scene are labeled for use by higher-level processes.

2.4.4 Error Analysis

In order to verify the accuracy of the system, and to help with its debugging, we plan to gather scene data using a new, *active* ranging system, developed by Jezouin at USC [28], which uses a pair of CCD cameras and a laser to find the distance from the left camera to the objects in the scene. The ranging system is expected to give ranges with the same 2-D resolution as the CCD cameras. In addition, the new ranging system provides a camera model which will be used to register the images. The laser is equipped with a servo- and galvo-controlled pair of mirrors which generates a plane of light which may be swept across a scene. The intersection of the light-plane with the objects in the scene is reflected as a thin vertical line across the faces of the objects in the scene.

This active system will generate a *ground-truth* for the scene that will be used during the error analysis to provide a quantitative analysis of how accurately we are able to determine object surfaces from the passive data alone.

3 Future Research

We are working toward a passive stereo image-processing system which extracts surfaces from matched features. Invariant features such as edges and corners are extracted from monocular images. Edges are found using the Nevatia-Babu Linear Feature Extractor [22] and matched using the Medioni Segment Matcher [4]. The resulting segment matches contain matching errors and the extracted features do not allow for the trivial extraction of disparity from the correct matches. This system will, therefore, post-process the matched segments using continuity and regional context (such as the gap-filling described in this paper) to detect and correct matching errors. Junctions, found by analysis of segment joins and endpoints, will be matched using the segment matches and continuity in the pair of images. Hypotheses are formed about missing and erroneous lines and junctions which will be acted upon if verified during later processing.

To accurately match edgels in order to reduce ranging error as well as remove artificial jaggedness in depth, we feel that it is necessary to postulate the surfaces of the underlying scene and therefore produce a dense description of that scene. Therefore, surfaces will be fit to *vertical* lines. This last stage will be performed in two steps, first to obtain an approximate surface sketch, which will be used to determine the location of limb edges and to propagate disparity information across the *horizontal* lines.

The principle contributions that are or will be made by this system are:

1. The use of stereo matching to drive monocular image processing.
2. A new classification of intensity junctions which provides an additional monocular cue to junction valid-

ity.

3. A solution to the *contrast-reversal* problem.
4. A dense representation using passive techniques without making any "*flat-surface*" assumption.
5. Develop a $2\frac{1}{2}$-D sketch with curved surface-patches fit to an extended wire-frame model of the scene.

Progress has already been made on the first three of the above goals. In addition, in order to demonstrate our assertion that an accurate 3-D representation may be acquired through passive stereo we will obtain an actively-acquired set of range data which is registered with the passively-acquired intensity images. This data will be used to generate a quantitative measure of the system.

4 Acknowledgement

The author would like to thank Gerard Medioni, Ramakant Nevatia, and Keith Price for their support and comments on this work.

References

[1] A. R. de Saint Vincent. A 3D perception system for the mobile robot Hilare. In *IEEE International Conference on Robotics & Automation*, pages 1105–1111, IEEE Computer Society, April 7-10 1986. San Francisco, California.

[2] S. Tsuji, J. Zheng, and M. Asada. Stereo vision of a mobile robot: World constraints for image matching and interpretation. In *IEEE International Conference on Robotics & Automation*, pages 1594–1599, IEEE Computer Society, April 7-10 1986. San Francisco, California.

[3] N. Ayache and B. Faverjon. A fast stereovision matcher based on prediction and recursive verification of hypothesis. In *Proceedings of the third Workshop on computer Vision: Representation and Control*, pages 27–37, IEEE Computer Society, October 1985. Bellaire, Michigan.

[4] G. Medioni and R. Nevatia. Segment-based stereo matching. *Computer Vision, Graphics, and Image Processing*, 31:2–18, 1985.

[5] V. Milenkovic and T. Kanade. Trinocular vision using photometric and edge orientation constraints. In *Proceedings of the DARPA Image Understanding Workshop*, pages 163–175, December 1985. Miami Beach, Florida.

[6] R. Mohan. Error detection and correction for stereo. In *Proceedings of the DARPA Image Understanding Workshop*, pages 433–442, December 1985. Miami Beach, Florida.

[7] R. D. Arnold. *Automated Stereo Perception.* PhD thesis, Stanford University, Stanford, California, March 1983. Technical Report STAN-CS-83-961.

[8] W. Grimson. *From Images to Surfaces: A Computational Study of the Human Early Visual System. Artificial Intelligence*, The MIT Press, Cambridge, Massachusetts, 1981. Based on the author's thesis (Ph.D. – MIT).

[9] S. Barnard and W. Thompson. Disparity analysis of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2(4):333–340, July 1980.

[10] S. Barnard and M. Fischler. Computational stereo. *ACM Computing Surveys*, 14(4):553–572, December 1982.

[11] G. Medioni and Y. Yasumoto. *Corner Detection and Curve Representation Using Cubic B-Splines.* Technical Report ISG 111, University of Southern California, September 1985.

[12] M. Shah and R. Jain. Detecting time-varying corners. *Computer Vision, Graphics, and Image Processing*, 28:345–355, 1984.

[13] D. Langridge. Curve encoding and the detection of discontinuities. *Computer Graphics and Image Processing*, 20:58–71, 1982.

[14] R. Nevatia and A. Huertas. *Building Detection in Simple Scenes.* Final Technical Report ISG 110, University of Southern California, September 1985.

[15] G Falk. Interpretation of imperfect line data as a three-dimensional scene. *Artificial Intelligence*, 3:101–144, 1972.

[16] Y. Shirai. Recognition of real-world objects using edge cues. In A. Hanson and E. Riseman, editors, *Computer Vision Systems*, pages 353–362, Academic Press, New York, 1978.

[17] Y. Shirai. A context sensitive line finder for recognition of polyhedra. *Artificial Intelligence*, 4:95–119, 1973.

[18] R. Shapira and H. Freeman. Computer description of bodies bounded by quadric surfaces from a set of imperfect projections. *IEEE Transactions on Computers*, C-27(9):841–854, September 1978.

[19] R. Brooks. *Symbolic Reasoning Among 3-D Models and 2-D Images.* PhD thesis, Stanford, Stanford, California, June 1981. Report No. STAN-CS-81-861.

[20] M. Herman and T. Kanade. *The 3D MOSAIC Scene Understanding System: Incremental Reconstruction of 3D Scenes from Complex Images.* Technical Report CMU-CS-84-102, Carnegie-Mellon University, Pittsburgh, PA, February 1984.

[21] R. Ohlander, K. Price, and R. Reddy. Picture segmentation using a recursive splitting method. *Computer Graphics and Image Processing*, 8:313–333, 1978.

[22] R. Nevatia and K.R. Babu. Linear feature extraction and description. *Computer Graphics and Image Processing*, 13:257–269, June 1980.

[23] D. Lowe and T. Binford. The recovery of three-dimensional structure from image curves. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 7:320–326, May 1985.

[24] S. Cochran. Accurate determination of three dimensional surfaces from passive stereo. Unpublished, December 1986. Ph.D. Dissertation Proposal.

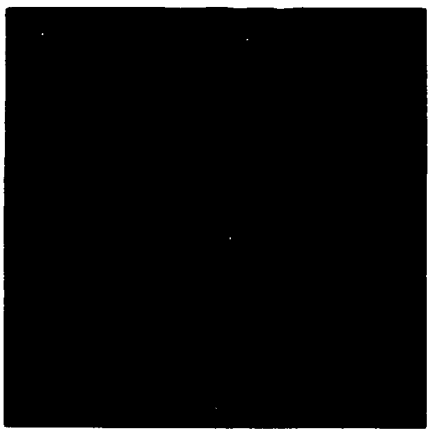
[25] A. Huertas and G. Medioni. Edge detection with subpixel precision. In *Proceedings of Computer Vision and Pattern Recognition Conference*, pages 633–636, IEEE Computer Society, June 19-23 1985. San Francisco, California.

[26] J. S. Chen, A. Huertas, and G. Medioni. Very fast convolution with laplacian-of-gaussian masks. In *Proceedings of Computer Vision and Pattern Recognition Conference*, pages 293–298, IEEE Computer Society, June 22-26 1986. Miami Beach, Florida.

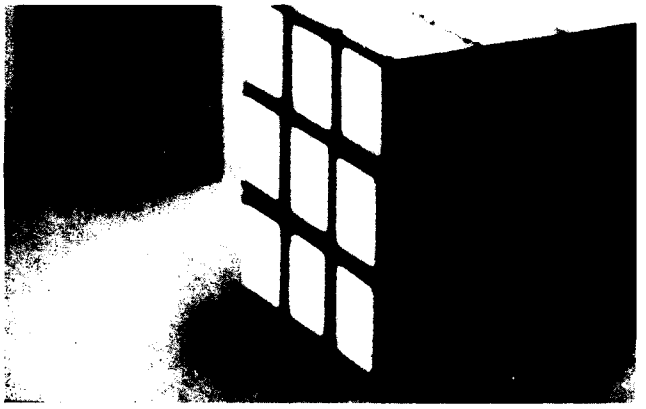
[27] D. Panton, C. Grosch, D. DeGryse, J. Ozils, A LaBonte, S. Kaufmann, and L. Kirvida. *Geometric Reference Studies.* Final Technical Report RADC-TR-81-182, December 1981. Volume 44, Number 12.

[28] G. Medioni and J. Jezouin. An implementation of an active stereo range finder. March 1987. To be published in the Proceedings of the Optical Society of America.

(a) Renault Part.

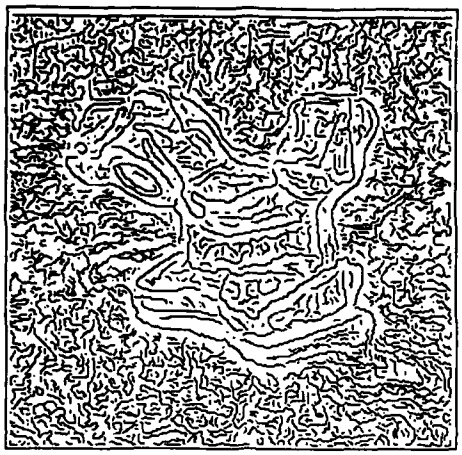


(b) Aerial View.

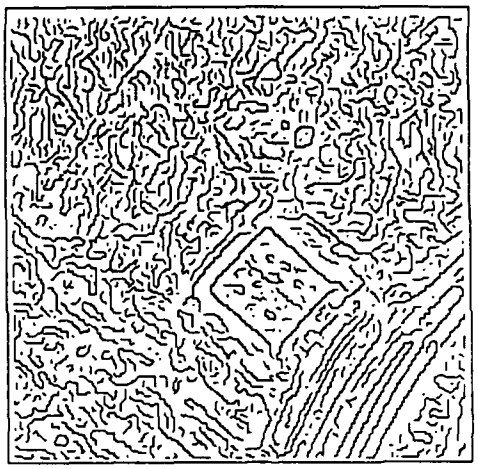


(c) Blocks.

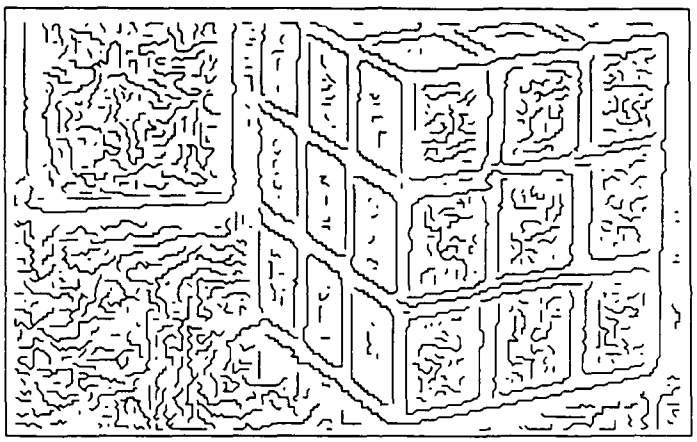
Figure 1: The Original Intensity Images



(a) Renault Part.



(b) Aerial View.



(c) Blocks.

Figure 2: Extracted and Linked Edgels.

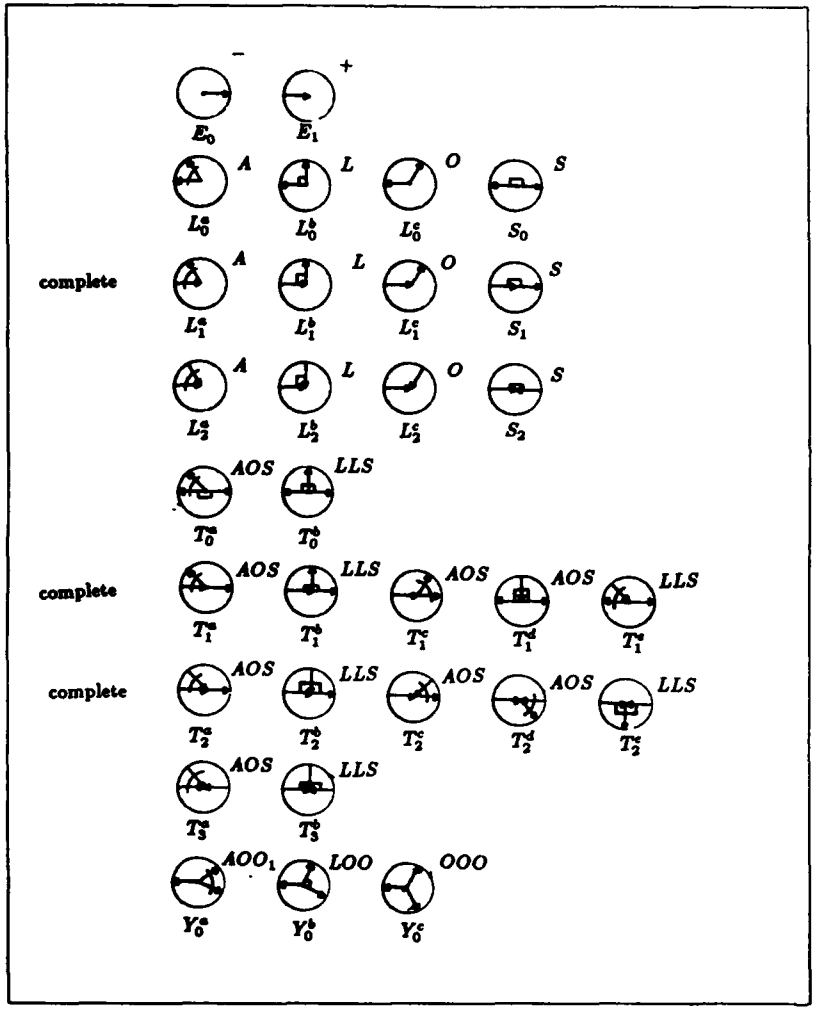
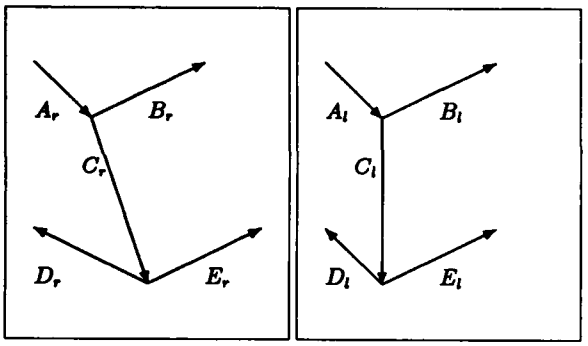


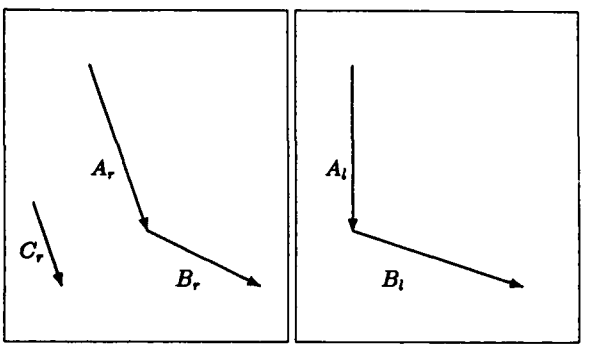
Figure 3: Possible Join Labels.



(a) Right Image. (b) Left Image.

$$\begin{array}{l} A_l \rightsquigarrow \{B_l, C_l\} \\ C_l \rightsquigarrow \{D_l, E_l\} \\ C_l \Leftrightarrow C_r \\ A_r \rightsquigarrow \{B_r, C_r\} \\ C_r \rightsquigarrow \{D_r, E_r\} \end{array} \Longrightarrow \begin{array}{l} A_l \Leftrightarrow A_r \\ B_l \Leftrightarrow B_r \\ D_l \Leftrightarrow D_r \\ E_l \Leftrightarrow E_r \end{array}$$

Figure 4: A Partial Match Which Can be propagated.



(a) Right Image. (b) Left Image.

$$\begin{array}{l} A_l \rightsquigarrow B_l \\ A_l \Leftrightarrow A_r \\ B_l \Leftrightarrow C_r \\ A_r \rightsquigarrow B_r \end{array} \Longrightarrow \begin{array}{l} B_l \Leftrightarrow B_r \\ C_r \not\Leftrightarrow \end{array}$$

Figure 5: A Bad Match Which Can be Resolved.

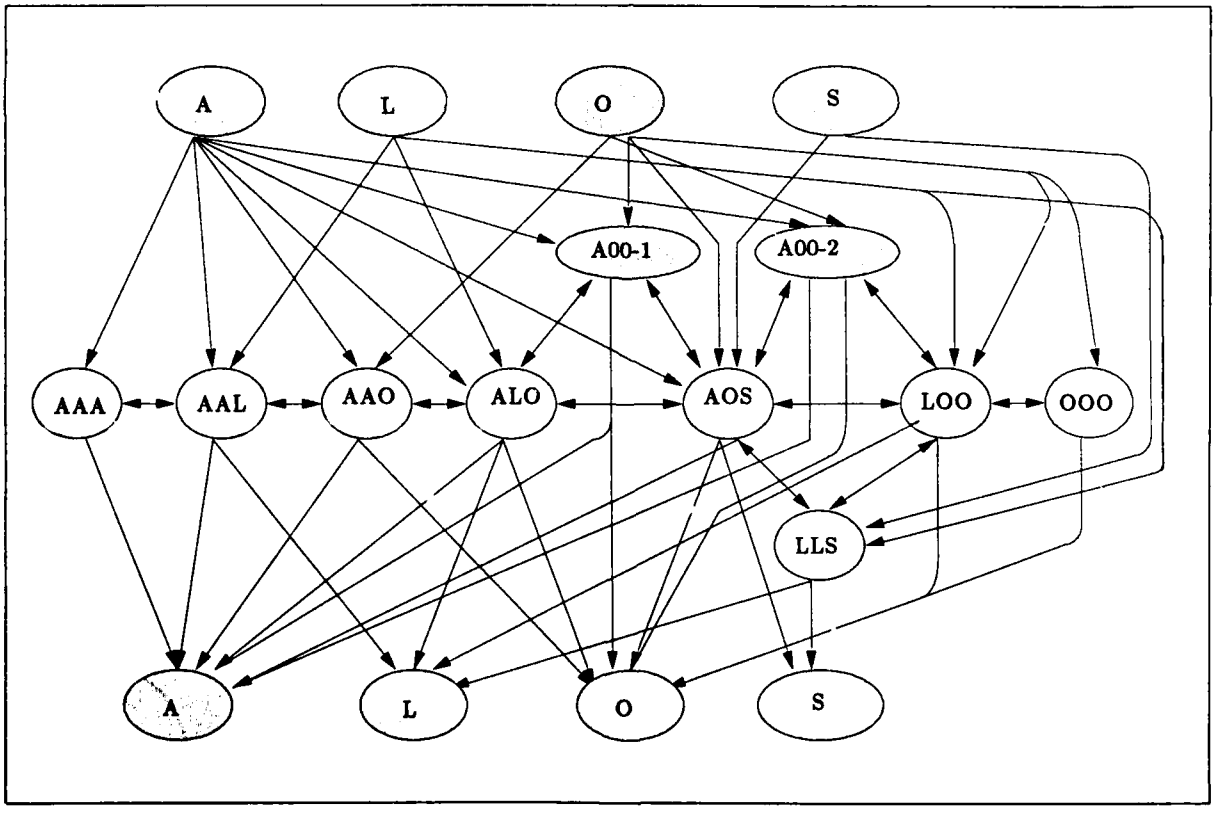


Figure 6: Junction Transition Graph. In addition to the the "ALOS" type, the corresponding segments must also approximately match.

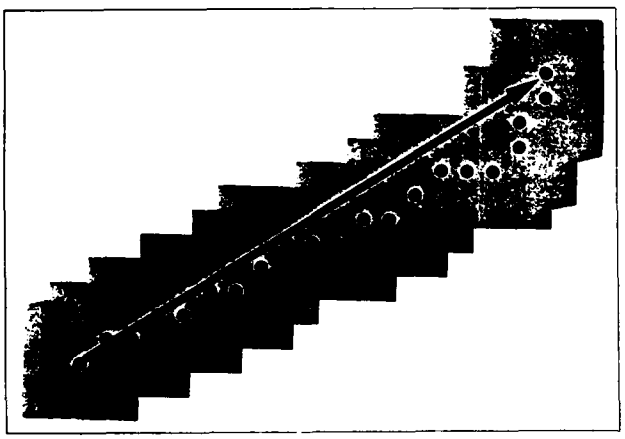


Figure 7: Search for Associated Regions.

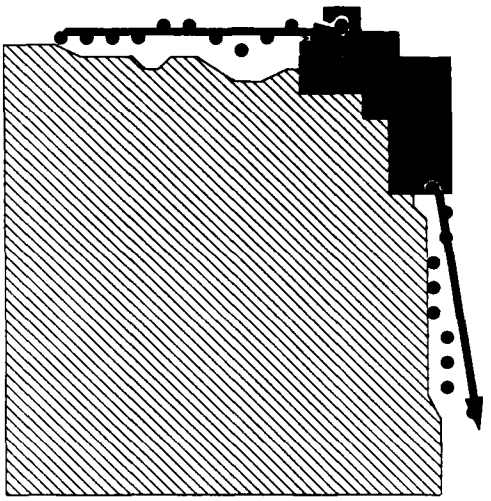
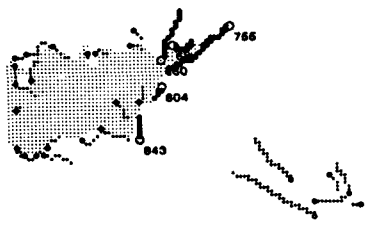


Figure 8: Search for Corner Endpoints.

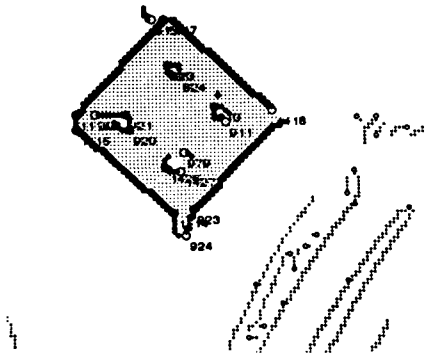
	A	L	O	S	AAA	AAL	AAO	ALO	AOO_1	AOS	LLS	AOO_2	LOO	OOO
A	0	1	2	3	1	1	1	1	1	1	2	1	2	3
L	1	0	1	2	2	1	2	1	2	2	1	2	1	2
O	2	1	0	1	3	2	1	1	1	1	2	1	1	1
S	3	2	1	0	4	3	2	2	2	1	1	2	2	2
AAA	1	2	3	4	0	1	2	3	4	4	5	5	5	6
AAL	1	1	2	3	1	0	1	2	3	3	4	4	4	5
AAO	1	2	1	2	2	1	0	1	2	2	3	3	3	4
ALO	1	1	1	2	3	2	1	0	1	1	2	2	3	4
AOO_1	1	2	1	2	4	3	2	1	0	1	2	2	3	4
AOS	1	2	1	1	4	3	2	1	1	0	1	1	2	3
LLS	2	1	2	1	5	4	3	2	2	1	0	2	1	2
AOO_2	1	2	1	2	5	4	3	2	2	1	2	0	1	2
LOO	2	1	1	2	5	4	3	3	3	2	1	1	0	1
OOO	3	2	1	2	6	5	4	4	4	3	2	2	1	0

Table 1: Junction Transition Distances (Assuming a unit transition cost).

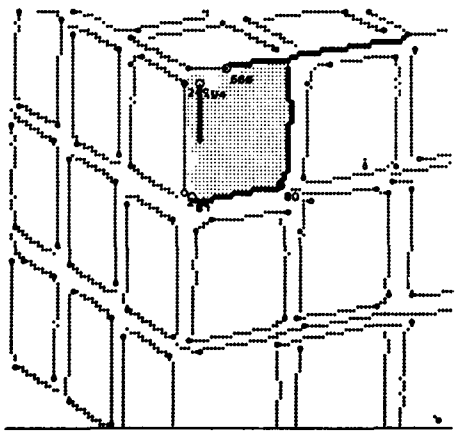
Figure 9: Segments Near Blob Boundary.



(a) Renault Part.

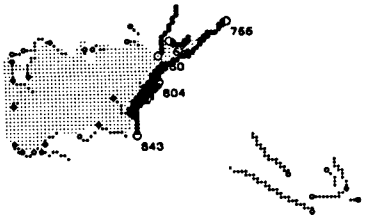


(b) Aerial View.

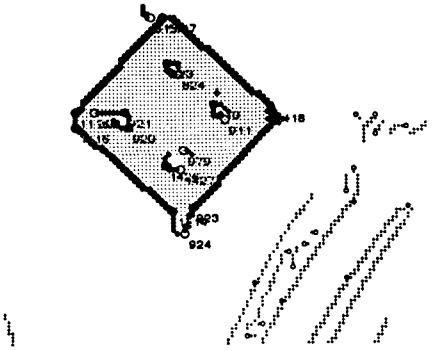


(c) Blocks.

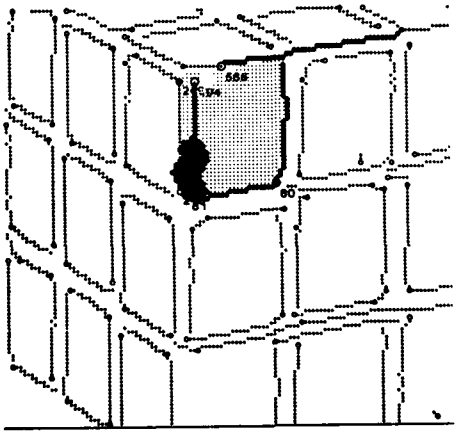
Figure 10: Results of Applying the Gap-Filling.



(a) Renault Part.



(b) Aerial View.



(c) Blocks.

STEREO MATCHING BY HIERARCHICAL, MICROCANONICAL ANNEALING

Stephen T. Barnard *

SRI International (Artificial Intelligence Center)
333 Ravenswood Avenue, Menlo Park, California 94025

Abstract

An improved stochastic stereo-matching algorithm is presented. It incorporates two substantial modifications to an earlier version: (1) a new variation of simulated annealing that is faster, simpler, and more controllable than the conventional "heat-bath" version, and (2) a hierarchical, coarse-to-fine-resolution control structure. The same Hamiltonian is minimized, but far more efficiently. The basis of microcanonical annealing is the Creutz algorithm . Unlike its counterpart, the familiar Metropolis algorithm, the Creutz algorithm simulates a thermally isolated system at equilibrium. The hierarchical control structure, together with a Brownian state-transition function, tracks grounds states across scale, beginning with small, coarsely-coded levels. Results for a 512x512 pair with 50 pixels of disparity are shown.

1 INTRODUCTION

Computational theories of vision often involve optimization, usually in an enormous state space and over a non-convex objective function. Recently, a subtle technique from statistical physics has been used for such computations [1,2,3,4]. It is based on the physical analogy of annealing a system of molecules to its ground state, and hence is called *simulated annealing* [5,6]. To grow a perfect crystal, one starts at a high temperature and then gradually cools the substance, staying as close to equilibrium as practical. This is directly analogous to what happens in simulated annealing: the Metropolis algorithm [7] is used to bring a synthetic system to equilibrium, and the macroscopic parameter *temperature* is used to control the rate of cooling.

The general theme is as follows:

- A particular (usually low-level) vision problem is considered.
- A representation is chosen in which the problem is modeled as an analog to a physical system of discrete "molecules." Each molecule typically corresponds to a location on a pixel lattice. The system has many degrees of freedom.
- An energy function (a Hamiltonian) is chosen that expresses constraints inherent in the problem. The solution is characterized by the *ground states* of the system; that is, those states that have the lowest energy. The ground states are difficult to specify because the state space of the system is huge — exponential in the number of molecules.
- The dynamics are simulated in a way that brings the system to the desired ground state, or at least to a very low-energy state that approximates a ground state.

This paper presents such a model for matching stereo images. An early version is described in [3]. Two major extensions have been made that permit substantially improved performance. First, a new variety of simulated annealing is employed. It uses an simpler alternative to the standard Metropolis algorithm which is more efficient, more easily implemented, and offers more control over the annealing process. Secondly, by representing the stereo pair as a Laplacian pyramid, the system is extended to operate over several levels of resolution. It exploits relatively quickly-computed minima at lower levels of resolution to initialize its state at higher levels. This method leads both to more efficiency and to the ability to deal with much larger ranges of disparities.

The basic representation remains unchanged. The state of the system encodes a dense map of discrete horizontal disparities, defined over the left image, which specify corresponding points in the right image. In the improved design, this state is relative to a particular level of resolution.

The energy of a lattice site is composed of two terms, each of which expresses a contraint important in stereo matching:

$$E_{ij} = |I_L(i,j) - I_R(i,j + D(i,j))| + \lambda|\nabla D(i,j)|$$

I_L and I_R are the left and right image intensities, or some simple function of them (such as a Laplacian). Subscripts i and j range over all sites in the left image lattice. D is the disparity map. The first term is the absolute difference in intensity between corresponding points (where the correspondences are defined by the current state of the system), and the second is proportional to the local spatial variation in the current state (the magnitude of the gradient of the disparity map). The constant λ is used to balance the terms.[1] This equation expresses two competing constraints: (1) the image intensities of corresponding points should be more-or-less equal, and (2) the disparity map should be more-or-less continuous.

The Hamiltonian,

$$E = \sum E_{ij} ,$$

Support for this work was provided by the Defense Advanced Research Projects Agency under contracts DCA76-85-C-0004 and MDA903-86-C-0084.

[1] The performance of the system is not very sensitive to the value chosen for λ. In the example of Section 4, as well as all three examples in [3], $\lambda = 5$ was used.

is unchanged; therefore, the ground states that we seek to determine or to approximate are the same. The new design is strictly concerned with improved performance. The techniques used to obtain it are not restricted to stereo matching, and should be considered for any simulated-annealing approach to low-level vision problems.

2 MICROCANONICAL ANNEALING

The conventional simulated annealing algorithm uses an adaptation of the Metropolis algorithm to bring a system to equilibrium at decreasing temperatures. The Metropolis algorithm defines a Markov process that generates a sequence of states, such that the probability of occurrence of any particular state is proportional to its Boltzman weight:

$$P(S) \propto \exp(-\beta E(S)) ,$$

where $E(S)$ is the energy of state S and β is the inverse temperature of the system. This process generates samples from the *canonical ensemble*; that is, the system is considered to be immersed in a heat bath with a controllable temperature. Annealing is accomplished by imposing a *schedule* for reducing the temperature, $1/\beta$, so as to keep the system close to equilibrium.

Creutz has described an alternative technique that simulates the *microcanonical ensemble* [8]. In this method, the total energy remains constant (for some fixed point in the schedule). Instead of simulating a system immersed in a heat bath, the Creutz algorithm simulates a thermally isolated system in which energy is conserved. It performs a random walk through state space, constraining states to a surface of constant energy. The simplest way to accomplish this is to augment the representation with an additional degree of freedom, called a demon, that carries a variable amount of energy, E_D. The total energy of the system is now:

$$E = E(S) + E_D .$$

Normally, the demon is constrained to have non-negative energy, although the possibility of giving it negative energy is useful in annealing, as will be discussed below.

In the Metropolis algorithm a potential new state S' is chosen randomly, and is accepted or rejected based on the change in energy:

$$\Delta E = E(S') - E(S) .$$

If ΔE is negative, the new state is accepted; otherwise, it is accepted with probability $\exp(-\beta \Delta E)$.

The Creutz algorithm is quite similar. If ΔE is negative, the new state is accepted, and the demon energy is increased ($E_D \leftarrow E_D - \Delta E$). If ΔE is non-negative, however, acceptance of the new state is contingent upon E_D: if $\Delta E < E_D$ the change is accepted, and the demon energy is decreased ($E_D \leftarrow E_D - \Delta E$); otherwise, the new state is rejected. Clearly, the total energy of the system remains constant.

The Creutz algorithm has several advantages. Unlike the Metropolis algorithm, it does not require the evaluation of transcendental functions. Of course, in practice these functions can be stored as tables, but we would like our algorithm to be adaptable to fine-grained parallel processors such as the Connection Machine. The small amount of local memory in such machines makes lookup tables unattractive. The Creutz algorithm can easily be implemented with only integer arithmetic — again, a significant advantage for fine-grained parallel processors and for VLSI implementation. Experiments indicate that the Creutz method can be programmed to run an order of magnitude faster than the conventional Metropolis method for discrete systems [9]. A further important advantage is that microcanonical simulation does not require high quality random numbers.

In conventional simulated annealing we control the process by specifying the temperature. In the microcanonical version, however, temperature is not a control parameter; it is a statistical feature of the system. In fact, standard arguments can be used to show that at equilibrium the demon energies have a Boltzman distribution:

$$P(E_D) \propto \exp(-\beta E_D) .$$

The inverse temperature can be determined from the mean value of the demon energy:

$$\beta = \frac{\ln(1 + 4/ < E_D >)}{4} .$$

Control of microcanonical annealing is accomplished by periodically removing energy from the system. The method used to generate the results in Section 4 is as follows:

1. Assume that the process begins in a random state S_0 of high energy with respect to the ground state. Call this $E_0 = E(S_0)$. The initial demon energy is zero.

2. Remove a fixed proportion of this energy, δE, by reducing the demon energy ($E_D \leftarrow E_D - \delta E$). The results of Section 4 were obtained with $\delta E = E_0/300$.

3. Run the Creutz algorithm until the system reaches equilibrium. A reasonable test for equilibrium is to consider the rate of accepted moves to states of higher energy. If the system is large enough, this rate will increase steadily until it approximates the rate of moves to states of lower energy. We terminate the process for a particular energy level when the rate of accepted moves to higher energy states decreases (measured over N site visits.

4. Repeat steps (2) and (3) until no further improvement is observed.

This procedure has only one free parameter: the ratio $\delta E/E_D$. While $E_D < 0$ the Creutz method operates as a "greedy" algorithm, accepting only moves to lower energy states and increasing E_D. When E_D becomes positive (which happens quickly if δE is small), the demon begins to exchange energy between lattice positions. As the ground state is approached, E_D remains negative because it cannot absorb more energy from the lattice.

If sites are visited in a regular scan, we observe an undesireable effect: after E_D is made negative, energy is removed from only a small area of the lattice. For example, if we visit the sites along scan lines from left-to-right, bottom-to-top, the algorithm will make "greedy" moves on the lower part of the lattice, resulting in significantly lower energy density in this area compared to the rest of the lattice. Many more scans may be required for this energy gradient to diffuse thoughout the entire lattice. This problem is easily overcome by visiting sites in random order.

The algorithm described above is sequential, and therefore is not suitable for parallel processing. Each local state transition can change E_D, which in turn can affect the next transition. Fortunately, the technique can be modified to a parallel one

by using a separate demon for each lattice site. (As we add demons, the technique moves toward a canonical-ensemble simulation. In fact, if the number of demons is very large compared to the number of sites, the technique specializes to the Metropolis algorithm [8].) Preliminary experiments with one demon per site indicate good performance, although the results of Section 4 were generated with the single-demon algorithm.

3 HIERARCHICAL ANNEALING

In the original model [3], annealing was performed only at the level of resolution of the stereo images. In some cases the images were first bandpass filtered to remove low-frequency components — in effect, a simple photometric correction for inconsistent sensor gains or film development. Lower and upper bounds on disparity were specified in advance, and all state transitions were considered with equal probability. As the range of disparity became large, this scheme required much larger amounts of computation. If we have m permissible disparities and N pixels, the size of the state space is m^N. If we double the range of disparity, the size of the state space increases by a factor of 2^N. Since N is a rather large number (2^{18} in the example shown in Section 4), we see that the state space grows explosively with increasing disparity range.

A natural extension of the method is to adopt a hierarchical, coarse-to-fine control structure. At a coarse level of resolution the number of lattice sites (i.e., the number of pixels) and the range of disparity are small, and therefore the size of the state space is relatively small.[2] We should be able to compute an approximate ground state quickly, and then use it to initialize the annealing process at the next, finer level of resolution.

Coarse-to-fine techniques have been widely used for image matching. For example, Moravec [10] used a resolution hierarchy to match discrete, point-like features. The stereo model described by Marr and Poggio [11] and further developed by Grimson [12] matched zero-crossings between hierarchies of bandpassed images. More recently, Witkin *et. al.* [13] have used a continuation method to minimize an energy function through scale space. In each case, the results of low-resolution matching were used to guide the system at higher resolutions.

The Laplacian pyramid, developed originally as a compact image coding technique [14], offers an efficient representation for hierarchical annealing. In a Laplacian pyramid, an image is transformed into a sequence of bandpass filtered copies, $I^0, I^1, I^2, ..., I^n$, each of which is smaller that its predecessor by a factor of 1/2 in linear dimension (a factor of 1/4 in area), with the center frequency of the passband reduced by one octave. This transform can be computed efficiently by recursively applying a small generating kernel to create a Gaussian (low-passed) pyramid, and then differencing successive low-passed images to construct the Laplacian pyramid.

After constructing Laplacian pyramids from the original stereo images, disparity is reduced by a factor of 1/2 in successive levels. Therefore, at some level, disparity is small everywhere. For typical stereo images, we can take this to be level $n - 4$. (For example, if the original images were a power of 2 in linear dimension, the Laplacian images at level $n - 4$ would be 16x16 pixels. Disparities in the range of 0 to 63 pixels in a pair of 512x512 images would be reduced to the range of 0 to 1 pixel, with truncation, at the $(n-4)$th level) We shall start annealing at this level, find an approximate ground state, at then expand the solution to the next level. To make this coarse-to-fine strategy work, however, we need two further modifications. We must use a different process for generating state transitions, and we must specify how a low-resolution result is used to start the annealing process at the next higher level.

In the original model, the probability of choosing a new disparity for consideration as a new state was uniformly distributed over the prior range of disparities:

$$P(d_j \rightarrow d_k) = \frac{1}{m} .$$

This is not compatible with our intention of guiding the process with lower-resolution results, however. We are now assuming the disparity of a lattice site to be close to its correct value, A more effective generating process is to restrict the disparities to increase or decrease by one pixel:

$$P(d_j \rightarrow d_k) = \begin{cases} .5 & \text{if } |d_j - d_k| = 1 \\ 0 & \text{otherwise} \end{cases} ,$$

with the further restriction that the disparities are not allowed to specify corresponding points outside the boundary of the right image. In this scheme the system undergoes Brownian motion through state space. An additional feature of this state-transition function is that it is no longer necessary to specify bounds on disparity in advance.

Expanding a low-resolution result to the next level is a little tricky. Obviously, one should begin by simply doubling the size of the low-resolution lattice, and also doubling the disparity values. Having done this, however, the new state has an artificially low energy because every odd disparity value is "unoccupied," and the new map is therefore more uniform that it should be. A spurious symmetry is imposed on the new state that is solely due to the quantization of the previous result, and that is likely to place the system near a metastable state (a local minimum) from which it cannot recover. Fortunately, there is an easy solution to this problem: break this symmetry by adding heat. One effective way is as follows:

1. Compute the energy E_0^k of the initial state at level k. (This state has been determined by doubling the result at level $k+1$.)

2. Add a fixed proportion of this energy, ΔE^k, by increasing the demon temperature ($E_D \leftarrow E_D + \Delta E^k$). The results of Section 4 were obtained with $\Delta E^k = E^k/10$.

3. Run the Creutz algorithm until the system reaches equilibrium.

4. Repeat steps (2) and (3) until the number of accepted higher-energy states exceeds the number of rejected higher-energy states.

4 RESULTS

Figures 1(a) and (b) are an aerial stereo pair (512x512 pixels each) covering part of Martin Marietta's test site for the Autonomous Land Vehicle Project near Denver. The terrain is dominated by a long, steep "hogback" (running diagonally

[2] Although the state space may be large in absolute terms.

across the middle of the images), that separates two broad valleys. The highest terrain is in the upper left. Several roads and a few buildings may be seen. Disparity ranges from zero to 50 pixels. Figures 1(c) and (d) show the Laplacian pyramids.

The results of applying the microcanonical, hierarchical annealing algorithm to this data are illustrated in Figure 2(a), which shows the sequence of approximate ground states found at each level of the pyramid. Disparity values are displayed in 15 greylevels with wraparound. At the coarsest level of resolution (the tiny 16x16 images), the system converged to a uniform disparity map. As the system descends through the resolution hierarchy, the detail of the approximate ground states becomes increasingly precise. The disparities along the right border of the map are incorrect because the left image does not overlap the right in this region, and pixels in this region therefore have no corresponding points. Figure 2(b) is a contour map of covering the upper left portion of the stereo pair, prepared by the Engineering Topographic Laboratory, and is included for comparison.

Figures 3 and 4 show some of the results in more detail. In Figure 3(a) a few points and columns have been selected from a magnified portion of the left image. The corresponding points in the right image are shown in Figure 3(b). Figure 4 is similar.

The largest problem solved by the previous version of the system was on the order of a 256x256 pair with 15 levels of disparity. This required about 12 hours on a Symbolics 3600. The result shown here is a 512x512 pair with 50 levels of disparity, and was computed in about the same amount of time. Consider the size of the state space of the two problems. The state space of the current problem[3] exceeds the size of the old problem by a factor of more than 10^{36000}.

5 CONCLUSIONS

Two major improvements to a stochastic stereo-matching system have been described: (1) a simulation of the microcanonical ensemble, as opposed to the canonical ensemble of conventional simulated annealing; and (2) the extension to a hierarchical control structure based on Laplacian pyramids. Both techniques are rather general in nature, and can probably be used in other simulated-annealing applications. Together, they permit solutions of stereo-matching problems far beyond the competence of the original system.

We are currently evaluating the quantitative increase in performance attributable to each new technique. Qualitatively, the use of microcanonical annealing yields perhaps an order of magnitude increase in efficiency. A potentially more important benefit is that it is a simpler computation than standard annealing, and is therefore more readily implemented in fine-grained parallel systems. The hierarchical control structure, combined with the Brownian state-transition function, contributes most of the increased performance.

References

[1] S. Geman and D. Geman, Stochastic relaxation, Gibbs distributions, and Bayesian restoration of images, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-6, n0. 6, November, 1984, pp. 721-741.

[2] P. Carnevali, L Coletti, and S. Patarnello, Image processing by simulated annealing, *IBM Journal of Research and Development*, vol. 26, no. 6, November 1985, pp. 569-579.

[3] S. Barnard, A stochastic approach to stereo vision, *Proc. 5th Nat. Conf. Artificial Intell.* (Philadelphia, Aug. 1986), vol. 1, American Association of Artificial Intelligence, Menlo Park, Calif., pp. 676-680.

[4] J. L. Marroquin, *Probabilistic solution of inverse problems*, Technical Report 860, MIT Artificial Intelligence Laboratory, September, 1985.

[5] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi, Optimization by simulated annealing, *Science*, vol. 220, no. 4598, May 13, 1983, pp. 671-680.

[6] S. Kirkpatrick and R.H. Swendsen, Statistical mechanics and disordered systems, *Comm. ACM*, vol. 28, no. 4, April 1985, pp. 363-373.

[7] N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, and E. Teller, Equations of state calculations by fast computing machines, *J. Chem. Phys.*, vol. 21, no. 6, June 1953, pp. 1087-1092.

[8] M. Creutz, Microcanonical Monte Carlo simulation, *Physical Review Letters*, vol. 50, no. 19, May 9, 1983, pp. 1411-1414.

[9] G. Bhanot, M. Creutz, and H. Neuberger, Microcanonical simulation of Ising systems, *Nuclear Physics*, B235[FS11], 1984, pp. 417-434.

[10] H. Moravec, Rover visual obstacle avoidance, *Proc. 7th Int. Jt. Conf. Artficial Intell.* (Vancouver, Canada, Aug. 1981), vol. 2, American Association of Artificial Intelligence, Menlo Park, Calif., pp. 785-790.

[11] D. Marr and T. Poggio, A theory of human stereo vision, Memo 451, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, Mass., Nov. 1977.

[12] W. E. L. Grimson, *From Images to Surfaces*, M.I.T. Press, Cambridge, Mass., 1981.

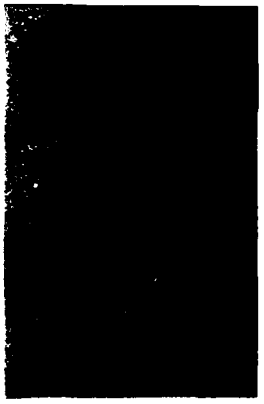
[13] A. Witkin, D. Terzopoulos, and M. Kass, Signal matching through scale space, *Proc. 5th Nat. Conf. Artificial Intell.* (Philadelphia, Aug. 1986), vol. 1, American Association of Artificial Intelligence, Menlo Park, Calif., pp. 714-719.

[14] P. Burt, The Laplacian pyramid as a compact image code, *IEEE Transactions on Communications*, vol. COM-31, no. 4, April 1983, pp. 532-540.

[3] Considering the range of disparity to be fixed at 50 pixels.

(a) Left image.

(b) Right image.

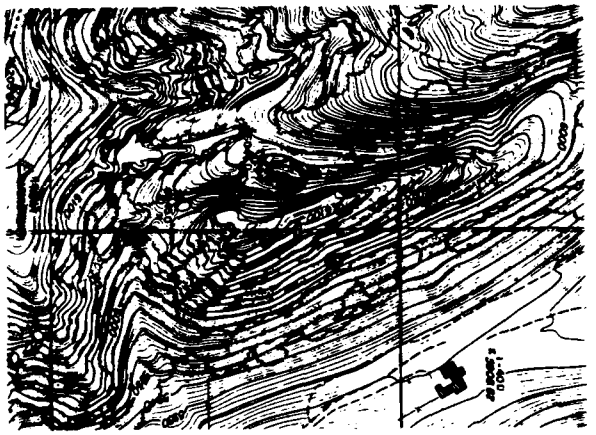


(c) Left Laplacian pyramid.

(d) Right Laplacian pyramid.

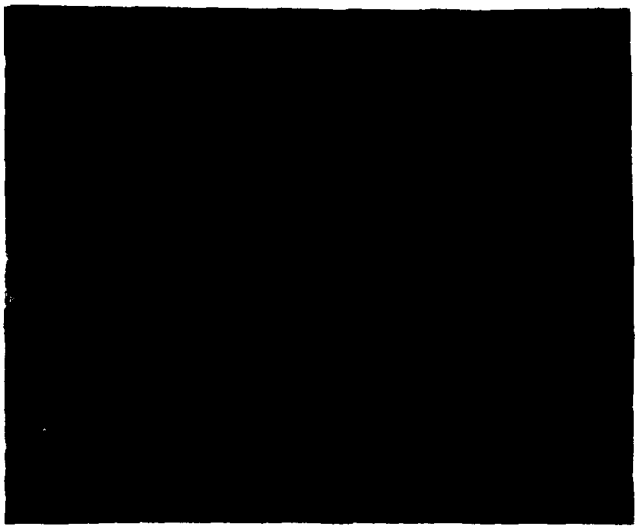
Figure 1: Aerial stereogram of the ALV test site.

(a) Pyramid of approximate ground states.

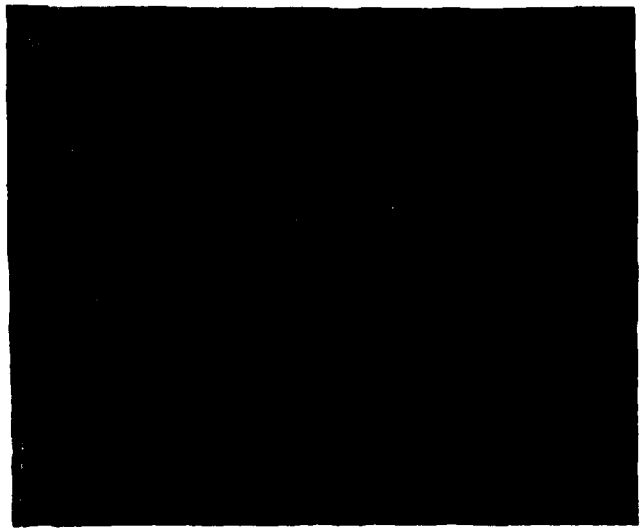


(b) Contour plot covering upper left portion.

Figure 2: Approximate ground states.

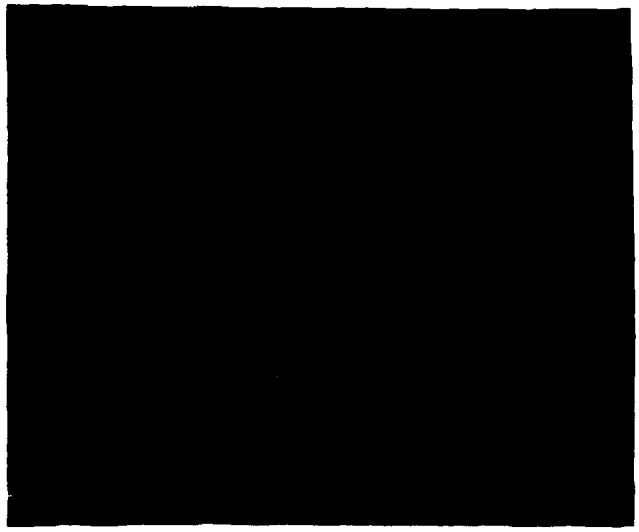


(a) Left image detail.

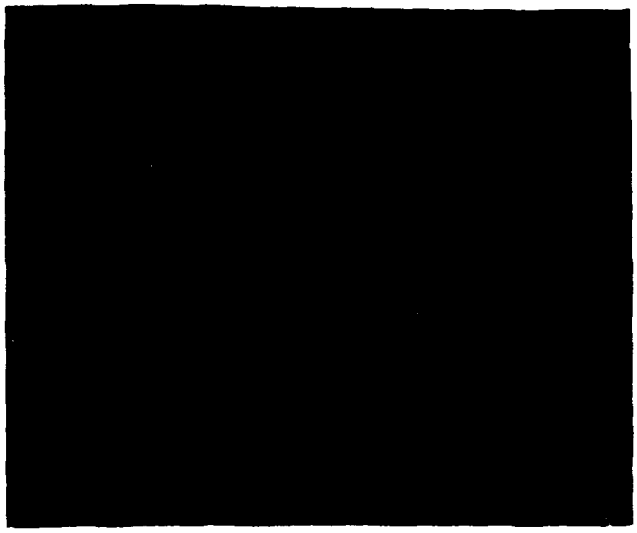


(b) Corresponding right image detail.

Figure 3: Detail A.



(a) Left image detail.



(b) Corresponding right image detail.

Figure 4: Detail B.

THE FORMATION OF PARTIAL 3D MODELS FROM 2D PROJECTIONS — AN APPLICATION OF ALGEBRAIC REASONING*

D. Cyrluk, D. Kapur, J.L. Mundy, and V. Nguyen

General Electric Corporate Research and Development
P.O. Box 8,
Schenectady, New York 12301

ABSTRACT

An algebraic model for three-dimensional objects is introduced. Algebraic deduction with the Gröbner basis is used to verify consistency between two object views. It is demonstrated that a useful model can be extracted from a single viewpoint using a new approach to scene labeling.

1. INTRODUCTION

Model-Based Recognition

Considerable research effort has been applied to the use of explicit three-dimensional models for object recognition [Besl and Jain]. These models have been frequently derived using conventional CAD tools [Roberts, Brooks, Lowe]. It is also possible to derive three-dimensional models from ideal line drawings [Markowsky and Wesley, Strat].

Models can also be derived from range data [Faugeras and Hebert, Grimson and Lozano-Perez, Connolly et al] or from stereo data [Faugeras et al., Mayhew]. In these cases, the models have been used to recognize objects as imaged in the same fashion. That is, a model of planar surfaces derived from range data is matched to planar surfaces derived from another range view of the object. Likewise, edge segments derived from stereo data are matched to edges derived from another stereo view of the object.

Constraints From a Single Projection

The focus of this paper is the derivation of a partial model from a single two-dimensional perspective view of an object and the use of this model to recognize the object from another viewpoint. It is not possible to derive a complete, three-dimensional model from a single view without assuming a great deal about the relationships between object surfaces and edges. For example, a common assumption is that the object is a polyhedron with adjacent faces and edges perpendicular. With this assumption, it is possible to establish the three-dimensional structure of the visible surfaces from a single view [Herman].

The investigation here centers on the case where the only assumption made about the object is that it is a polyhedron. The initial experiments have assumed that the perspective image transformation can be approximated by an *affine* transformation. An affine transformation is an orthographic projection with subsequent scaling of coordinate axes. The affine approximation is more realistic than the usual orthographic assumption, but is more constraining than the full perspective case.

A polyhedron is a collection of three elements; the vertex, the edge, and the face. A vertex is a three-dimensional point and represents the intersection of two or more edges or three or more faces. An edge is bounded by two vertices and is the intersection of at least two faces. A face is a planar region bounded by a sequence of edges and vertices. The affine projection of a polyhedron produces a set of two-dimensional edges and vertices. The image plane is partitioned into a set of closed regions, which are bounded either by projections of edges and vertices or the image border.

The least restrictive constraint that can be derived from the projection is simply that the image vertices are related to the corresponding object vertices by the linear transformation equations where the three-dimensional coordinates of each vertex are unknown, along with the six affine transformation parameters. These constraints are not sufficient to usefully define the three-dimensional structure of the object. It is necessary to introduce a grouping of vertices, edges and faces of the projection to provide significant constraints on the three-dimensional object configuration.

One example of such a group is to identify a cycle of edges and vertices in the projection image that corresponds to a visible face of the three-dimensional polyhedron. The identification of such groups requires that a labeling operation be performed on the projection [Malik, Kapur et al, Nguyen]. The fact that the cycle elements must be coplanar can then be used to further constrain the three-dimensional structure. Excellent reviews on the nature of these projection and coplanarity constraints are available [Sugihara, Crapo].

Recognition as Algebraic Consistency

Once the set of constraints is established from a given view of the object, they can be used as a model for recognition. In the work described here, the model constraints are expressed as a set of algebraic equations in terms of unknown three-dimensional coordinates and transformation parameters. If we consider another two-dimensional projection, which is hypothesized to be another view of the object, then the equations derived from each projection should be algebraically consistent; of course, the correct assignment has to be made between the corresponding elements of each view.

*This work was supported in part by the DARPA Strategic Computing Vision Program in conjunction with the Army Engineer Topographic Laboratories under Contract No. DACA76-86-C-0007.

The use of feasible constraints for object recognition is motivated by the ACRONYM system [Brooks]. In the case of ACRONYM, the consistency between a known three-dimensional model and its projection in an image was tested using a form of the SUP-INF method of reasoning about linear inequalities. The inequalities arise because of tolerances on the expected transformation and image feature position. By contrast, in the work reported here, we do not have a three-dimensional model, but only the constraints imposed by a two-dimensional view. Another major difference is that the decision procedure used in the current work permits non-linear constraint equations, but not inequations (inequalities). Actually, Brooks could handle a limited form of nonlinearity associated with the trigonometric forms involved in rotation.

The process of recognition, in the current context, is to first form assignments between vertex groups in each projection and then to test the algebraic consistency of the resulting equations. The consistency is tested by determining if the *ideal* of the set of equations, taken as polynomials equal to zero, contains the unity element. Briefly, the ideal is a set of polynomials that are all of the linear combinations of polynomials from the original set. If the ideal contains the unity element, then the set of equations is inconsistent. These concepts will be discussed in detail later, but it can be seen that a unit element in the ideal can generate any polynominal as a linear combination. This condition is similar to that arising from an inconsistent set of axioms in logic, where any statement can be proven by reductio-ad-absurdum. The presence of unity in the ideal can be determined using a system of term rewriting rules, called the Gröbner basis [Kapur]. The Gröbner basis is generated from the original set of polynominals using a completion procedure involving the interaction of pairs of polynomials, expressed as rewrite rules.

If the equations are consistent, then the two views correspond to the same three-dimensional object; or at least they share some solutions for the possible objects that are consistent with both projections. The consistency also allows a more specific determination of the three-dimensional configuration of the object. For example, if the transformation between the views is given in advance, then the correspondence between equations is similar to the feature matching done in classical stereo analysis. The matching between vertices provides a disparity or depth value for each vertex match.

If the transformation between views is unknown, then the depths cannot be determined, but only constrained by the assignment between a vertex from each view. It is reasonable to refer to this case as *algebraic stereo*; the object surface depth is not explicitly determined but the *variety* (set of equation zeroes) of the equations represents a space of possible object surfaces. In any case, the introduction of new constraints, either from hypotheses about geometric constraints on groups of projection elements, or from new views of the object, will reduce the number of unknown coordinate values.

The remainder of the paper will give details about the constraints, the analysis of consistency and some examples of experiments.

2. IMAGES OF POLYHEDRA

The Viewing Transformation

The situation of interest to us here is illustrated in Figure 2.1. A polyhedral object is being viewed from two or more viewpoints. To simplify the discussion we assume that the object coordinate frame is the same as that of the first view point. There is a, possibly unknown, coordinate transformation between the two viewpoint locations. The three-dimensional geometry and topological structure of the polyhedron is not known. All that is available are the two-dimensional projections of the object in each viewplane, π and π'. The goal is to determine if the two projections actually correspond to the same polyhedron.

First let us consider the transformation between the three-dimensional vertex locations and their corresponding two-dimensional position in the image plane. The natural transformation that results from a standard imaging system is the perspective transformation. However, it is not essential to use this exact relationship in most practical situations. It can be shown that an approximation to perspective, the affine transformation, is quite appropriate for most viewing situations [Thompson and Mundy].

The form of the affine transformation is given by the following matrix relation:

$$\mathbf{p} = w\,[\mathbf{I}_2][\mathbf{R}]\,\mathbf{P} + \mathbf{p}_0$$

where w is an affine scale factor and $\mathbf{R}$ represents a rotation matrix for the orientation of the object reference frame relative to the image reference frame. The matrix $[\mathbf{I}_2]$ indicates the projection from three dimensions into the two-dimensional image plane,

$$\mathbf{I}_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

$\mathbf{p}$ is the location of the projected vertex position vector, $\mathbf{P}$, in the image plane. The two-dimensional vector, $\mathbf{p}_0$, represents a translation in the image plane.

The rotation matrix, $\mathbf{R}$, can be represented as the product of three matrices,

$$\mathbf{R} = [\mathbf{R}_z][\mathbf{R}_y][\mathbf{R}_x]$$

which represent rotations about each of the view plane coordinate axes. For example,

$$\mathbf{R}_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \mathrm{Cos}\phi & \mathrm{Sin}\phi \\ 0 & -\mathrm{Sin}\phi & \mathrm{Cos}\phi \end{bmatrix}$$

where ϕ is the angle of rotation about the x axis. The complete transformation thus requires six parameters: three rotations, two translation components and a scale factor.

Without loss of generality, we assume that the projection of the object into the first viewplane involves no rotation, translation and a unity scale factor. The only unknowns are the depths (z component) of the three-dimensional object vertices. The x,y components of vertex $\mathbf{P}$ are just the vertex projection in image coordinates.

The affine viewing projection and the transformation between views are combined into a single transformation between corresponding image vertex locations in the two views. That is,

$$\mathbf{p}' = w\,[\mathbf{I}_2][\mathbf{R}]\,\mathbf{P} + \mathbf{p}'_0$$

where $\mathbf{p}'$ is the location of the projection of vertex $\mathbf{P}$ in the second view. In this case, $\mathbf{R}$ represents the rotation between image reference frames. The scale factor, w, arises from differences in image depths between views. $\mathbf{p}'_0$ is the component of the translation vector between viewpoints which lies in the second viewplane, π'.

It should be noted that our procedure assumes the location of projected image vertices to be exact. This assumption is not appropriate when dealing with actual intensity image data. Feature locations can be at least several pixels in error for actual images. Ultimately, this problem must be solved, but the main focus of this paper is to explore the formal properties of geometric and topological constraints in image projections.

The Topology of Projections

In addition to the geometric relationships just discussed, it will be important to consider the relationship between the topology of the three-dimensional polyhedral surface and the topology of the corresponding projection. We observe that a polyhedral image is a segmentation of the image plane into open and connected regions, joined by vertices and edges. We call these regions, faces of the image, and define a vertex-edge-face topology in the image space similar to the one for 3D polyhedra. Using point set topology [Henle], vertices, edges, faces of a polyhedral image are defined as follows:

vertex - *A vertex is a point of the image plane which is the intersection of at least two edges of the image. The angular sectors between consecutive edges around the vertex must be part of distinct faces of the image. So a vertex is not only a connection of distinct edges, but also a junction of distinct faces of the polyhedral image.*

edge - *An edge is a segment of the image plane which connects two distinct faces of the image. The edge is closed iff its two end points are both vertices of the image. The edge is open iff only one of its end points is a vertex of the image. In this case, the edge must be cut off by the image boundary.*

cycle - *A cycle is a closed chain of vertices and edges. A cycle* **C** *is an out-cycle (resp. in-cycle) of a face* **F**, *iff the direction of the cycle* **C** *is clockwise (resp. counter-clockwise), as we walk along* **C** *with the face* **F** *on the right hand side. Since all edge of the cycle is bounded by two vertices in the cycle, the cycle is a closed set of the image plane.*

face - *A face is a connected subset of the image plane locally bounded by chains of edges, called a boundary. A face is closed iff its boundary includes one out-cycle, zero or many in-cycles. A face is open iff its boundary is not closed, or it is only bounded from the interior by in-cycles. In this case, the face must be cut off by the image border.*

With the point set topology in the image space defined above, the set union of all topological sets (vertices, edges, and faces) of the image is equal to the open set of the image plane strictly inside the image boundary. Note that no point of the image is strictly inside two distinct topological sets. So mathematically, the segmentation of the image plane can be represented by a planar network of nodes. The nodes are not only the vertices as in [Huffman, Clowes, Waltz], but also the edges and faces of the image. The polyhedral image is represented more explicitly by a planar network of nodes linked between themselves by adjacency links. This planar network leads to a parallel implementation of image labeling.

For now, our polyhedral world excludes Origami objects that have folds of planar surfaces. So, the junctions in the image come either from occlusion, or from projection of 3D polyhedral vertices. We use Huffman-Clowes junctions for occlusion and trihedral vertices [Huffman, Clowes], and Malik's Fundamental Junction Equation for multihedral vertices [Malik]. However, junctions at vertices are not the only possible local constraints in labeling. We introduce two other labeling constraints around edges and face boundaries of the image, described respectively by junction-pairs, and junction-loops. These extensions of the labeling system provide a unified body of constraints over the vertex, edge, and face components of a projection.

Since a face is an open and connected region, its boundaries must be disjoint. We can label the boundaries independently of each other. The labeling constraints of these boundaries are represented by n-tuples of junctions, called junction-loops. A junction-loop is a consistent labeling of the chain or cycle of vertices and edges. Previously, the labeling consistency over arbitrary open chains of vertices and edges is captured by Waltz's filtering process, [Waltz]. Waltz and others are forced to do depth-first search for global labelings to capture the labeling consistency over boundary cycles. We also must do a depth-first search to enumerate all the junction-loops for a face boundary, however our depth-first search is local to a face, rather than global to the whole image.

Similarly, a junction-pair is a consistent local labeling of an edge. The junction-pair is a more complete description of the edge than the edge-label. Junctions, junction-pairs, and junction-loops are local labelings of a node consistent with itself and all its adjacent neighbors, Figure 3.1. Just as faces joined by vertices and edges completely represent the image segmentation, the junctions, junction-pairs, and junction-loops completely describe the local labeling constraints between the nodes in the image.

Nodes also have labels. For example, vertices are labeled V, Y, A, T, or M, respectively, for L, Fork, Arrow, T, or Multihedral junctions. Edges are labeled +, −, or >, respectively, for convex, concave, or occluding. Faces are labeled B, V, or O, respectively, for background, visible, or partially occluding face. Labels and local labelings describe *all* the labeling constraints at a node, and are organized into hierarchies.

Labels and labelings at a node are related by the subset relation between hierarchical classes. The hierarchies and the subset relation lead to formal definitions of constraint satisfaction between two adjacent nodes. The image topology leads to a uniform constraint propagation over all vertex-edge, edge-face, and face-vertex links. Labeling of a polyhedral image is equivalent to constraint satisfaction and propagation (CSP) over all nodes in the image. Each node has local constraints described by labels and labelings. Each node only interacts with its adjacent neighbors. The result of CSP is local consistencies or inconsistencies at all the nodes in the network.

Nguyen proved by induction that globally consistent labelings exist if and only if all the nodes in the network are local-

ly consistent. The induction proof confirms that the labels and labelings at a node are necessary and sufficient local constraints for the labeling of an image. The output is a network of nodes, with labels and local labelings attached to each node. This represents all locally/globally consistent labelings of the polyhedral image.

Figure 3.2 traces the parallel labeling of two blocks, one on top of the other. The parallel labeling starts with the input image, and default labels for all the nodes (i.e. vertices, edges, and faces) in the image, frame 1. Then, it finds all local labeling constraints at all vertices, face boundaries, and edges of the image, described respectively by junctions, junction-loops, and junction-pairs, frames 2 to 4. Constraint satisfaction and propagation is done uniformly at all the nodes in the image, from every node to its neighbors, through vertex-edge, edge-face, and face-vertex links. Frames 5 and 6 describe the result of CSP. Attached to each node is the number of local labelings.

The face surrounding the two blocks has 16 local labelings, corresponding to the blocks floating in air, or resting against some imaginary surface at some of its bounding edges, frame 5. The two interpretations correspond to the face labeled as image background (B) or as polyhedral face (F). Note that the blocks sitting on top of a horizontal surface can be thought as the blocks floating in air, and infinitesimally touching the surface. The surrounding face is a touchable background, and is labeled B. The detection of the background face surrounding the blocks further restricts the labeling of the blocks, without changing their shape interpretations, frames 7 and 8.

Labeling identifies significant groups of vertices which can be used to derive algebraic constraints. For example, all the vertices in a visible face (labeled V) are related by coplanarity equations, frame 8 of Figure 3.2. Locally, vertices connected by convex or concave edges (labeled respectively $+$, $-$) are on the planes of the adjacent faces. A vertex at a T occlusion only belongs to the occluding face. Occlusion gives an inequality relating the depths of the vertices on the same line of sight, but lying on different faces of the 3D scene [Sugihara]. Currently, inequalities are not handled and we rely only on coplanarity and projection constraints. The next section will describe the algebraic form of these constraints in detail.

3. ALGEBRAIC CONSTRAINTS

As indicated above, we view an object model to be a system of algebraic equations. These equations come from various sources:

1. The affine transformation of vertices.
2. The constraints on vertex groups.

   One such vertex group is the face. The constraints associated with the face is the coplanarity of the face's vertices.

The above constraints correspond to constraints derived from a single image. Additionally, we want to determine whether an assignment of vertices between vertex groups of two images is consistent. These constraints are introduced by equating vertex variables between the two images.

The basic idea behind our approach is to use these constraints to form maximally consistent assignment sets. A maximally consistent assignment set is a consistent set of vertex assignments such that the addition of any additional vertex assignment would cause inconsistency. The set of equations associated with a maximally consistent assignment set forms a model for the object.

A set of constraints may be consistent simply because there are too many degrees of freedom. Such a set of constraints is of little use in further refining a model. Thus, when we use a vertex group to constrain a set of assignments, we want to make sure that its constraints are sufficiently strong to either cause inconsistency, or, in case of a consistent assignment, to allow us to significantly refine the model. Thus it is natural for us to ask to what extent does a vertex group refine our 3D model and allow us to detect inconsistency. We will pursue these questions in this section.

We start with the affine transformation equations for a vertex group consisting of n vertices. These equations were presented earlier. As indicated there are two equations per vertex, one each for x and y. Thus, if we are trying to determine the consistency of two images, we will have a total of $4n$ equations, $2n$ for each image. Of course these equations are not necessarily independent of each other. As for the unknowns, there are six parameters for each transformation, the three angles, the two translation parameters, and the scaling factor. Additionally, the x, y, and z components of each object vertex are unknown, giving an additional $3n$ unknowns. Thus if we just consider the equations generated from the affine transformation, we have $4n$ equations, and $3n+12$ unknowns. If, with no loss in generality, we assume that the object coordinate frame is the same as the coordinate the frame for one of the images (that is, its transformation is simply identity), then we have $2n$ equations and $n+6$ unknowns. Thus with no other constraints we need at least six vertices in a vertex group to detect an inconsistent assignment.

Now suppose that the vertext group is additionally constrained to lie in a plane. The equation describing coplanarity is: $ax+by+cz+d=0$, where a, b, c, and d are new variables which determine a plane, and (x,y,z) is a vertex in the object. For each point in the vertex group there will be one plane equation. Thus for a vertex group representing a face with n vertices there will be $3n$ equations and $n+10$ unknowns. Thus with no other constraints we would need at least five vertices in a vertex group to detect an inconsistent assignment.

Note that a similar analysis can be found in [Sugihara], who also represents constraints as sets of equations. A major difference between our approaches is the manner in which under-constrained equations are handled. Sugihara deals with the problem by introducing cost functions for intensity and texture properties, and then uses a minimization procedure to select from the set of under-constrained solutions a "best" solution. In our approach we try to infer as much as possible from the available constraints by solving the set of equations symbolically. An advantage of Sugihara's method is that he can deal with inequalities and, to some extent, imprecise data. A main advantage of our approach is that a set of algebraic equations can naturally represent a set of multiple solutions to an under-constrained problem; i.e., we are not forced to select a specific solution.

We now describe our symbolic approach to testing consistency.

4. TESTING CONSISTENCY

Given a set of equations that describe the affine projection, topological properties, and geometric properties for a vertex group, we must be able to solve the following problems:

1. Is an assignment of vertices in two projections consistent?
2. Given a consistent assignment, what can we conclude about the transformation parameters and depth of the object?
3. Given two vertex groups with consistent assignments, can they be merged in a consistent manner?

The approach to these problems is the same approach used in [Kapur] to prove geometry theorems. The main idea is that a set of assignments is consistent if and only if the equations describing the assignments have a common zero. The remaining problems can be solved as a by-product of testing the consistency of a set of equations.

Algebraic Foundation

As indicated above, the main problem is to determine whether a set of equations have a common zero. In this section we give a precise formal description of this problem.

Given the field Q of rationals, let $Q[x_1, ..., x_n]$ be the set of all polynomials with variables $x_1, ..., x_n$, and rational coefficients. We will be considering solutions of equations in $Q[x_1, ..., x_n]$ belonging to the set of complex numbers C.

Definitions: Let $p \in Q[x_1, ..., x_n]$. The equation $p = 0$ is *consistent* iff there exist $v_1, ..., v_n \in C$, such that $p(v_1, ..., v_n)$, the result of substituting $v_1, ..., v_n$ for $x_1, ..., x_n$, respectively, in p evaluates to 0. $(v_1, ..., v_n)$ is a *zero* of p in C^n. The equation $p = 0$ is *inconsistent* otherwise. Given a set $S \subset Q[x_1, ..., x_n]$, S is *consistent* iff there exist $v_1, ..., v_n \in C$, such that for every $p \in S$, $p(v_1, ..., v_n)$ evaluates to 0. The set S is *inconsistent* otherwise.

Given two images, I1 and I2, let S1 be the set of equations generated from I1, and S2 be the set of equations generated from I2, then the problem of testing whether a set of vertex assignments, VA, is consistent is almost equivalent to the problem of testing whether the set of equations $T = S1 \cup S2 \cup VA$ is consistent.

The two problems are not exactly identical since a set of polynomial equations are consistent even if the only common zeros are complex, whereas in this case the set of vertex assignments would not be physically realizable. There are other approaches that avoid this problem (see for example [Arnon]), but they are considerably more computationally expensive than our approach.

Hilbert's Nullstellensatz

Hilbert's Nullstellensatz [van der Waerden] allows us to determine whether a set of equations is consistent.
Let $p \in Q[x_1, ..., x_n]$ that evaluates to zero at all the common zeros of $p_1, ..., p_i$ in the n-dimensional affine space of C, then $p^q = h_1p_1 + h_2p_2 + ... + h_ip_i$ for some natural number q and polynomials $h_1, h_2, ..., h_i \in Q[x_1, ..., x_n]$.

A special case of the above theorem is the following important corollary: If $p_1, ..., p_i$ have no common zeros in the n-dimensional affine space of C, then 1 belongs to the ideal generated by $p_1, ..., p_i$.

Thus the problem of testing whether a set of vertex assignments is consistent is reduced to the problem of testing whether the ideal generated by a set of polynomials contains 1. Our method of solving this problem uses the Gröbner basis computation described next.

The Gröbner Basis

An important problem in ideal theory is the ideal membership problem. Given a set of polynomials, S, let (S) denote the ideal generated by S. The set S is called a *basis* of the ideal (S). Given a polynomial, p, and a basis S, the ideal membership problem is to determine whether $p \in (S)$. [Buchberger] invented the Gröbner basis to solve this and other problems in polynomial ideal theory. Intuitively a Gröbner basis, G, is a set of polynomials that have the property that when used as rewrite rules they will reduce any polynomial in (G) to 0.

Polynomials as Rewrite Rules

The main idea behind the Gröbner basis is to view a set of polynomials as a rewrite system. A Gröbner basis is then a finitely terminating, confluent rewrite system. Before proceeding further we give some general background on rewrite systems.

A *rewrite rule* is a pair (l,r), usually written $l \rightarrow r$. l is called the left hand side, r the right hand side. A rewrite rule can rewrite an "object" if some part of the object matches the left hand side of the rule. For example the string rewriting rule, $a \rightarrow b$, can rewrite the string ab to bb. A rewrite system is a collection of rewrite rules. If an object cannot be rewritten any further then the object is said to be in *normal form*. We use the notation $s \rightarrow_R t$, to mean that s can be rewritten to t in one step using rewrite system R. $s \rightarrow^*_R t$ means that s can be rewritten in arbitrarily many steps to t.

Two important properties of rewrite systems are finite termination and confluence. **Definition:** $\rightarrow$ is *finitely terminating* (sometimes called *noetherian*) iff there does not exist an infinite sequence $t_1 \rightarrow t_2 \rightarrow ...$. For example, the rewrite system:

$a \rightarrow b$
$b \rightarrow a$

is not finitely terminating since the string a can be rewritten indefinitely.

Definition: $\rightarrow$ is *confluent* iff
$\forall t_1, t_2, t_3 \, (t_1 \rightarrow^* t_2 \wedge t_1 \rightarrow^* t_3) \Rightarrow \exists t_4 \, (t_2 \rightarrow^* t_4 \wedge t_3 \rightarrow^* t_4)$.

Intuitively, this means that if an object can be rewritten in two different ways to different objects then the two different objects can eventually be rewritten back to a common object. If a rewrite system is both finitely terminating and confluent then every object has a unique normal form. For example, the rewrite system:

$a \rightarrow b$
$ab \rightarrow b$

is not confluent since the string ab has two normal forms, namely bb and b. By adding the rule $bb \rightarrow b$, the above rewrite system can be made confluent.

We now show how a set of polynomials can be viewed as a rewrite system, and how from a finite set of polynomials we can generate a finitely terminating, confluent rewrite system, i.e. a Gröbner basis. We first show how a polynomial can be viewed as a rewrite rule.

Given a polynomial ring, $Q[x_1, ..., x_n]$, with indeterminates $x_1, ..., x_n$, and coefficients from Q, we define a term as a sequence of indeterminates or 1. A monomial is a term with a coefficient from Q. The simplified sum-of-products form of a polynomial is a sequence of monomials all with distinct terms. By using associativity, commutativity, and distributivity any polynomial can be put into simplified sum-of-products form.

For example, given the polynomial $(x_1x_2 + x_2 + x_2)(x_1)$, its simplified sum-of-products form is $x_1x_1x_2 + 2x_1x_2$. This polynomial will usually be written as $x_1^2x_2 + 2x_1x_2$.

We want to turn a set of polynomials into a finitely terminating rewrite system. The general method of ensuring that a rewrite system is finitely terminating is to determine a well founded ordering, $>$, on polynomials and to make sure that for every rewrite rule $l \rightarrow r, l > r$. Thus in order to use polynomials as rewrite rules we must first define a well founded ordering on polynomials.

We start by first defining an ordering on the indeterminates: $x_1 < x_2 < ... < x_n$. This ordering induces a total ordering on terms in many different ways, two of which are

1. Total degree ordering

   In this ordering we define for a term, t, deg(t) = the number of indeterminates in t. For example, $deg(x_1^2x_3) = 3$, and $deg(1) = 0$. Terms are compared in the following way:

   i. if $deg(t_1) < deg(t_2)$ then $t_1 < t_2$

   ii. if $deg(t_1) = deg(t_2)$ then compare t_1 and t_2 using lexicographic ordering.

   For example $x_2 < x_1^2$ since $deg(x_2) < deg(x_1^2)$.

2. Lexicographic ordering

   In this ordering terms are only compared lexicographically, the degree of a term is not taken into account. For example $x_1^2 < x_2$, since x_1^2 is lexicographically smaller than x_2.

The ordering that we choose to use depends on the application we have in mind. In general, the degree ordering is preferable if we are just trying to detect contradiction, while the lexicographic ordering is preferable if we want to solve a system of equations. This will be discussed further later.

In any case both orderings can be extended to monomials and then to polynomials. If $m_1 = c_1t_1$, and $m_2 = c_2t_2$, where c_1 and c_2 are coefficients and t_1 and t_2 are terms then $m_1 < m_2$ iff $t_1 < t_2$.

Let p_1, and p_2 be two polynomials such that $p_1 = m_1 + m_2 + ... + m_k$, and $p_2 = h_1 + h_2 + ... + h_l$, both in simplified sum-of-products form, with $m_1 > m_2 > ... > m_k$, and $h_1 > h_2 > ... > h_l$. Then $p_1 > p_2$ iff $\exists i < l (m_i > h_i \wedge \forall j < i (m_j = h_j))$. It is easy to see that this totally orders polynomials.

In polynomial p_1, above, m_1 is called the head monomial, and p_1 can be transformed to the following rewrite rule: $m_1 \rightarrow -m_2 - ... - m_k$. This rewrite rule is finitely terminating since under our ordering m_1 is greater than the polynomial $-m_2 - ... - m_k$. Note also that by dividing the entire polynomial by c_1 we can always form rewrite rules whose left hand side's coefficient is 1.

Now that we have a way of ensuring finite termination, we want to be able to ensure confluence. Intuitively, the way confluence is ensured is to look at the polynomial rewrite system and try to determine the possible ways that it could be used to rewrite a polynomial in more than one way. This is done by taking two rules and from their left hand sides forming a new term that can be rewritten by both rules. For example, given the following two rules:

1. $x_1x_2 \rightarrow r_1$ (r_1 is this rule's right hand side, whatever it may be.)
2. $x_2x_3 \rightarrow r_2$

If we take the least common multiple (lcm) of the two left hand sides we get $x_1x_2x_3$, which can be rewritten by both rules. Using rule 1 it rewrites to $x_3 * r_1$. Using rule 2 it rewrites to $x_1 * r_2$. In order to ensure confluence it must be the case that the polynomial, $x_3 * r_1 - x_1 * r_2$, reduces to 0. This polynomial is called an *s-polynomial* of rule 1 and rule 2, and we say that rule 1 and rule 2 overlap. If with the current set of rules this polynomial does not reduce to 0 then we can force it to reduce to 0 by transforming its reduced form to a rewrite rule and adding it to the current system. This is the basic step of the Gröbner basis computation, and is repeated until all s-polynomials reduce to 0. Additionally, it is useful to use the new polynomial to reduce the existing polynomials, hopefully causing previous polynomials to reduce to 0. Note that the polynomials that can be reduced using the new rule have to be removed from the current rule set and formed into new rules.

Here is an outline of the Gröbner basis algorithm:

Input: F, a finite set of polynomials.
Output: G, a Gröbner basis, such that (G) = (F).

```
1  G := ∅; pairset := ∅; new_polys := F
2  loop until 1 ∈ G or (pairset = ∅ and new_polys = ∅)
3    loop while new_polys ≠ ∅
4      p := a polynomial from new_polys
5      new_polys := new_polys - {p}
6      G := G ∪ {p}
7      add to pairset pairs of polynomials with p and
         polynomials in G
8    (p1,p2) := pair from pairset
9    sp := normal form of s-polynomial(p1,p2)
10   remove from G any polynomials that are reduceable by
       sp, remove their corresponding pairs from from
       pairset, and add these polynomials to new_polys
11 return G
```

Buchberger showed the correctness and termination of this algorithm [Buchberger].

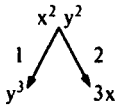
We will illustrate the algorithm with a few examples.

Example 1:

Let $F = \{x^2y - y^2, xy^2 - 3\}$, with $x > y$. Turning these equations into rewrite rules using total degree ordering yield:

1. $x^2y \rightarrow y^2$
2. $xy^2 \rightarrow 3$

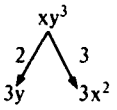
Rules 1 and 2 yield the following s-polynomial:



Thus yielding rule:

3. $y^3 \rightarrow 3x$

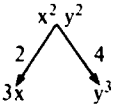
Rules 2 and 3 overlap in the following manner:



Thus yielding rule:

4. $x^2 \rightarrow y$

This new rule simplifies rule 1 to 0, thus deleting it. Rules 2 and 4 overlap:



This results in the polynomial $y^3 - 3x$, which can be reduced to 0 by rule 3. Rules 3 and 4 do not have to be overlapped since they do not have any variables in common, and the resulting s-polynomial would trivially be reducable to 0 using rules 3 and 4.

Since there are no pairs that we have not considered, rules 2, 3, and 4 form a Gröbner basis for F.

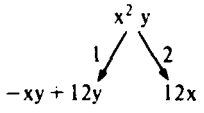
Example 2:

This example illustrates how the Gröbner basis can be used to detect inconsistency:

Let $F = \{ \ + x - 12, xy - 12, y^2 - 4\}$ with $x > y$. Turning these equations into rewrite rules using total degree ordering yield:

1. $x^2 \rightarrow -y + 12$
2. $xy \rightarrow 12$
3. $y^2 \rightarrow 4$

Rules 1 and 2 overlap:



After reduction this results in the rule:

4. $x \rightarrow y - 1$

Which can be used along with rule 3 to reduce rule 2 to:

5. $y \rightarrow 8$

Rule 5 then reduces rule 3 to:

6. $1 \rightarrow 0$.

and we have a contradiction, showing that the original set of polynomials do not have common zero.

Example 3:

This third example illustrates a major drawback of this approach: the following equations do not have a common real zero, but they do have a common complex zero, thus their Gröbner basis does not contain 1.

Let $F = \{x^2 + 1\}$. This polynomial is its own Gröbner basis, and yet it does not have a real zero.

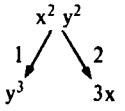
Example 4:

This example illustrates how the solution to a set of equations can be found using lexicographic order.

Let $F = \{x^{2y} - y^2, xy^2 - 3\}$, with $x > y$, as in example 1. Turning these equations into rewrite rules using lexicographic ordering yield:

1. $x^2 y \rightarrow y^2$
2. $xy^2 \rightarrow 3$

Rules 1 and 2 yield the following s-polynomial:



Thus yielding rule:

3. $x \rightarrow 1/3y^3$

Note that this rule is oriented differently than in example 1. Rule 3 can then be used to simplify both rules 1 and 2. It turns that both these rules simplify to

4. $y^5 \rightarrow 9$

Thus the final Gröbner basis consists of the polynomials:

1. $y^5 - 9$
2. $x - 1/3y^3$

Note that the first polynomial is in terms of only y, while the second is in terms of x and y. Thus, solving the first equation for y and plugging in this value in the second equation yields the solution:

$y = 3^{2/5}$ and $x = 3^{1/5}$

In general, running the Gröbner basis computation using lexicographic ordering generates triangular sets of equations like the one in this example.

We now give examples of the use of the Gröbner basis in solving our image understanding problem.

5. EXAMPLES

In this section we illustrate the application of our method on a very simple object. We defined a 20 by 20 by 20 cube centered about the origin and formed two orthographic projections of it along two diagonals. The resulting images are shown in Figure 5.1. The first image corresponds to a rotation of 45 degrees around the x-axis, followed by a rotation of −35.26 degrees around the y-axis, followed by a rotation of −30 degrees about the z-axis. The second image corresponds to a rotation of −45 degrees around the x-axis, followed by a rotation of −35.26 degrees around the y-axis, followed by a rotation of 30 degrees about the z-axis. From these projections the x and y components of the two images were calculated. Since these are real numbers and the Gröbner basis method requires exact coordinates, these numbers had to be represented by using polynomial equa-

tions. For example, if the x component of a point is $200^{1/2}$, then it would be input as $x^2 - 200 = 0$, instead of as $x = 14.142137$. Note that the Gröbner basis does not allow us to distinguish between $+200^{1/2}$ and $-200^{1/2}$. However, if we have two points x_1 and x_2, one whose value is $+200^{1/2}$ and the other whose value is $-200^{1/2}$, then these can be represented by the two equations, $x_1^2 - 200 = 0$, and $x_1 + x_2 = 0$.

We performed several experiments testing algebraic consistency on this cube. These experiments were performed using an implementation of the Gröbner basis algorithm running on the Symbolics lisp machine. These experiments are described below:

Experiment 1: Determining depth given the transformation between the two views.

In this experiment we took the face corresponding to vertices 1, 2, 3, and 4 in Figure 5.1 as our vertex group. The set of equations corresponding to this problem consists of the union of the following equations:

1. The trig identity.

   This set consists of the equations of the form $\cos^2\psi + \sin^2\psi = 1$, one for each of the six rotation angles (three per image). Note that the angles themselves are not actually the indeterminates in the problem. The actual indeterminates are the sine and cosine of the angles. Thus for angle, ψ, there would correspond two variables, cos_psi, and sin_psi.

2. The specification of the transformation between the two images.

   To simplify the computation we used the coordinate system of the first image as the object coordinate system. This meant that the three angles, ψ_1, θ_1, and ϕ_1, corresponding to rotation about the x, y, and z axis, respectively, are all 0. This is specified by the equations: cos_psi_1 − 1 = 0, and sin_psi_1 = 0, etc.

   The transformation from image 1 to image 2 is given by $\psi_2 = -70.53$, $\theta_2 = 0$, and $\phi_2 = 0$. Since $\cos(-70.53) = 1/3$, and $\sin(-70.53) = 8/9^{1/2}$, the equations for ψ_2 are 3cos_psi_2 − 1 = 0, and 9sin_psi_2^2 − 8 = 0.

   Note that this set of equations makes the trig identity equations redundant.

3. The specification of the image points in terms of the object points.

   As indicated earlier, the general equation for the points in the second image in terms of the first image is

   $$\mathbf{p}_2 = w[\mathbf{I}_2][\mathbf{R}_z][\mathbf{R}_y][\mathbf{R}_x][\mathbf{p}_1] + \mathbf{p}'_0$$

   In this example w is 1, and p'_0 is 0. Thus only the rotation matrix is left. Instead of multiplying the three rotation matrices out we simply added intermediate variables to keep track of the values after the rotations about each axis. For example, the equations describing the rotation of the object point x_0 into the image point x, would be

   $$x_1 - x_0 = 0$$
   $$x_2 - (x\cos\theta + y\sin\theta) = 0$$
   $$x - (x\cos\phi - y\sin\phi) = 0$$

   In experiments it turned out that factoring the equations this way sped up the Gröbner basis computation. We have similar equations for the coordinates of each point in the image.

4. The coplanarity equations.

   For each point, (x, y, z) in a vertex group we would add the equation, $ax + by + cz + d = 0$. Since, in our example none of the cube's faces passes through the origin, we know that d is not 0. This lets us divide the equation by d, giving us a simpler equation, $ax + by + cz + 1 = 0$. We have an equation such as this for each point in the vertex group.

5. The vertex assignments.

   In order to equate point p in image 1 with object coordinates x, y, and z, with point p′ in image 2 with object coordinates x′, y′, and z′, we would add the three equations:

   $x' - x = 0$,

   $y' - y = 0$, and

   $z' - z = 0$.

   In this example we assigned vertex 1 in image 1 to vertex 1 in image 2, vertex 2 in image 1 to vertex 2 in image 2, and so on.

The union of all these equations made up the input to the Gröbner basis. In all there were 126 equations. (Some of these are shown in Figure 5.2.) However, many of them are redundant. After 272 seconds and 43 critical pairs the Gröbner basis was computed. It consisted of 119 polynomials, including the solution to the vertex depths.

Experiment 2: Determining inconsistency given the transformation between the two views.

This problem is very similar to the first problem. The only difference is in the assignment set. Instead of the consistent assignment given above, we gave it the inconsistent assignment:

$$v_{1,1} = v_{2,3}$$
$$v_{1,2} = v_{2,1}$$
$$v_{1,3} = v_{2,4}$$
$$v_{1,4} = v_{2,2}$$

where $v_{i,j}$ is vertex j in image i. After 48 seconds and 0 critical pairs the Gröbner basis detected inconsistency. Note that detecting inconsistency is much easier than determining consistency. This is because contradiction might be detected well before all possible pairs are examined.

Experiment 3: Computing the Gröbner basis without giving the transformation between views.

This problem is severely underconstrained. In fact, with only one face of the cube, any assignment is consistent. Given the same assignment as in experiment 1, the Gröbner basis computation ran for 1 hour, generating 1824 critical pairs. The depths and rotation parameters were still only partially constrained. Given the assignment in experiment 2, the lisp machine ran out of space before finding a Gröbner basis.

Experiment 4: Extending a consistent assignment with

another vertex group.

In this problem we took the Gröbner basis generated in the previous experiment and added to it the equations corresponding to the face containing vertices 3, 4, 5, 6 (see Figure 5.1). With a consistent assignment the Gröbner basis terminated after 10 hours, 22 minutes and 4432 critical pairs. With an inconsistent assignment the Gröbner basis detected inconsistency after 4 hours, 15 minutes and 2055 critical pairs.

Further Experiments:

We performed further experiments on the cube by modifying some of the experiments above to include a scaling factor, and a translation. As expected this resulted in longer running times for the Gröbner basis computation, but these problems still finished.

While the above experiments have been somewhat successful, they indicate a need to make progress on several fronts.

6. EXTENSIONS

The extensive computation required for the examples in the previous section clearly indicates a need to reduce the size of the Gröbner basis. This is crucial when we want to augment a completed Gröbner basis with additional vertex groups, or when we want to merge two Gröbner bases together.

In these cases we would like to be able to extract from the Gröbner basis only the information that is likely to cause inconsistency. For example, in experiment 1 above, many of the 119 polynomials in the final basis involve only the intermediate variables used to specify the image points in terms of the object points. Once the basis has been computed, these intermediate variables are no longer of any use. The information that will be useful are the equations involving the affine transformation parameters, and the unknown vertex depths.

This information can be isolated by using the lexicographic ordering instead of the degree ordering. However, running the Gröbner basis with lexicographic ordering can be considerably more expensive than using degree ordering. An alternative is to first generate a Gröbner basis using degree ordering, and then run the Gröbner basis computation on the resulting set of equations just long enough to generate the equations constraining the indeterminates that we are interested in.

Another line of attack would be to classify a *type* associated with small groups of equations. The type class would be determined by geometric and topological relations that define the group. If such a type function can be defined, then it would be possible to avoid generating elements of the Gröbner basis that can never lead to meaningful geometric deductions. That is, one would not attempt to generate critical pairs between rules of dissimilar type. One example of such a type class has already been discussed, the coplanar set; if we had another group, say three mutually perpendicular edges, it would not seem meaningful to generate critical pairs across these two equation sets.

Another major area of extension is the control mechanism for generating hypotheses and maintaining consistent assignment sets. One aspect is similar to truth maintenance systems, or TMS, where one maintains a network of dependencies between assumptions and conclusions. In our case the assumptions concern assignments between elements of the projections in the two views as well as assumptions that may be made about the object or the viewing transformation.

The other aspect is to use a hierarchy to define the control strategy. Assumptions about the object and viewing transformation may take the form of a hierarchy in which more specific assumptions are tried and relaxed until one finds the least general assumption that leads to consistency. For example, in the case of the viewing transformation, perspective is more general than affine which is more general than orthographic. Correspondences between views that are inconsistent under orthography may be consistent under perspective. The same type of hierarchy can apply to assumptions about the object; for example, a polyhedron is more general than a rectilinear solid, which is more general than a cube.

REFERENCES

[1] Arnon, D.S., Collins, G.E., and McCallum, S., "Cylindrical Algebraic Decomposition I: The Basic Algorithm," *SIAM J. of Computing 13*, pp. 865-877, 1984.

[2] Arnon, D.S., Collins, G.E., and McCallum, S., "Cylindrical Algebraic Decomposition II: An Adjacency Algorithm for the Plane," *SIAM J. of Computing 13*, pp. 878-889, 1984.

[3] P. Besl and R. Jain, "Three-Dimensional Object Recognition," *Computing Surveys 17*, (1), March 1985.

[4] Brooks, R.A., "Symbolic Reasoning Among 3D Models and 2D Images," *Artificial Intelligence 17*, p. 285, 1981.

[5] Buchberger, B., "An Algorithm for Finding a Basis for the Residue Class Ring of a Zero-dimensional Polynomial Ideal," (in German) Ph.D. Thesis, Univ. of Innsbruck, Austria, Math. Inst., 1965.

[6] Buchberger, B., "Gröbner Bases: An Algorithmic Method in Polynomial Ideal Theory," in: *Multidimensional Systems Theory*, N.K. Bose (ed.), pp. 184-232, Reidel, 1985.

[7] Clowes, M.B., "On Seeing Things," *Artificial Intelligence 2* (1), pp. 79-116, 1971.

[8] Connolly, C.I. and Stenstrom, J.R., "Construction of Polyhedral Models from Multiple Range Views," *Proc. 8th International Conference on Pattern Recognition*, 1986.

[9] Crapo, H. "Spatial Realizations of Linear Scenes," in *Structural Topology*, no. 13, 1986.

[10] Faugeras, O., Ayache, N., Faverjon, B., Lustman, F., "Building Visual Maps by Combining Noisy Stereo Measurements," *Proc. IEEE Conf. on Robotics and Automation*, p. 1433, 1986.

[11] Faugeras, O. and Hebert, M., "A 3D Recognition and Positioning Algorithm Using Geometrical Matching Between Primitive Surfaces," *Proc. 7th International Joint Conference on AI*, 1983.

[12] Grimson, E. and Lozano-Perez, T., "Search and Sensing Strategies for Recognition and Localization of

Two- and Three-Dimensional Objects," *Proc. 3rd International Symposium on Robotics Research*, 1985.

[13] Henle, M., *A Combinatorial Introduction to Topology*, Freeman and Co., San Francisco, 1979.

[14] Herman, M., *Representation and Incremental Construction of a Three-Dimensional Scene Model*, Carnegie-Mellon University Report, CMU-CS-85-103, 1985.

[15] Huffman, D.A., "Impossible Objects as Nonsense Sentences." *Machine Intelligence 6*, pp. 295-323, 1971.

[16] Kapur, D., "Geometry Theorem Proving Using Hilbert's Nullstellensatz," NSF Workshop on Geometric Reasoning, June 1986.

[17] Lowe, D., *Perceptual Organization and Visual Recognition*, Kluwer Academic Publishers, Boston Mass., 1985.

[18] Malik, J., "Interpreting Line Drawings of Curved Objects," PhD Thesis, Stanford University, 1985.

[19] Markowsky, G., Wesley, M.A., "Fleshing Out Wire Frames," *IBM Journal of Research and Development 24*, (5), 1980.

[20] Mayhew, J.E., et al., Keynote address, IEEE Conf. on Computer Vision and Pattern Recognition, June 1986.

[21] Nguyen, V., "A Parallel Algorithm for Labelling Polyhedral Images," submitted to IJCAI87, 1987.

[22] Roberts, L.G., "Machine Perception of Three-Dimensional Solids," *Optical and Electro-Optical Information Processing*, J.T. Tippett et al. (ed.), MIT Press, Cambridge, Mass., 1965.

[23] Strat, T., Spatial Reasoning From Line Drawings of Polyhedra, *Proc. IEEE 1984 Workshop on Computer Vision Representation and Control.*

[24] Sugihara, K. "An Algebraic Approach to Shape From Image Problems," *Artificial Intelligence 23*, p. 59, 1984.

[25] Thompson, D. and Mundy, J., "Three-Dimensional Model Matching From an Unconstrained Viewpoint," *Proc. IEEE Conf. on Robotics and Automation*, April 1987.

[26] van der Waerden, B.L., *Modern Algebra*, Vol. I and II, Frederick Ungar Publishing Co., New York, 1966.

[27] Waltz, D.L., "Generating Semantic Descriptions from Drawings of Scenes with Shadows," *The Psychology of Computer Vision*, P.H. Winston (ed), McGraw Hill, New York, 1972.

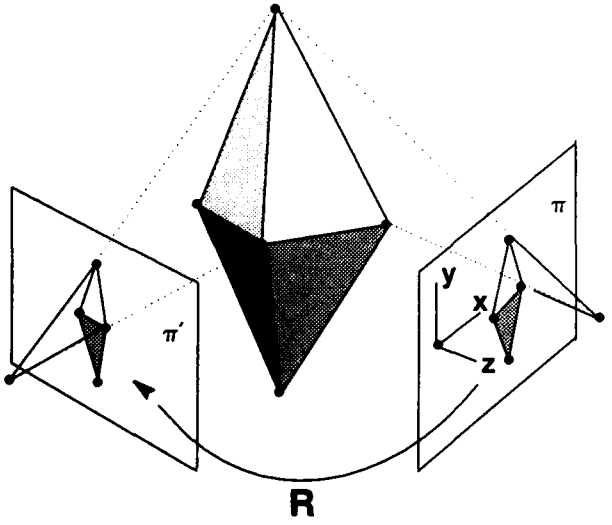


Figure 2.1. A polyhedral object and two viewing reference systems.

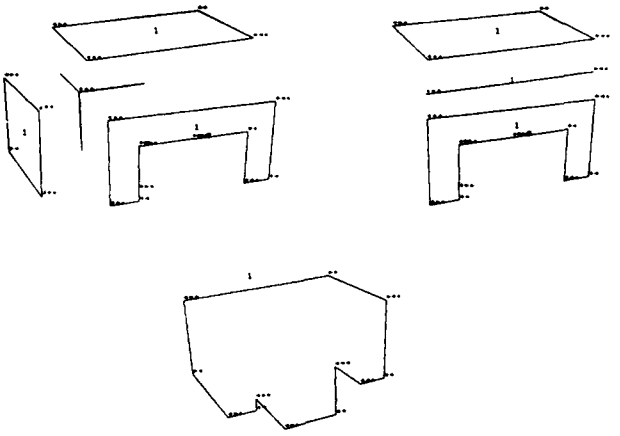


Figure 3.1. Local labelings at a node: junction, junction-pair, junction-loop.

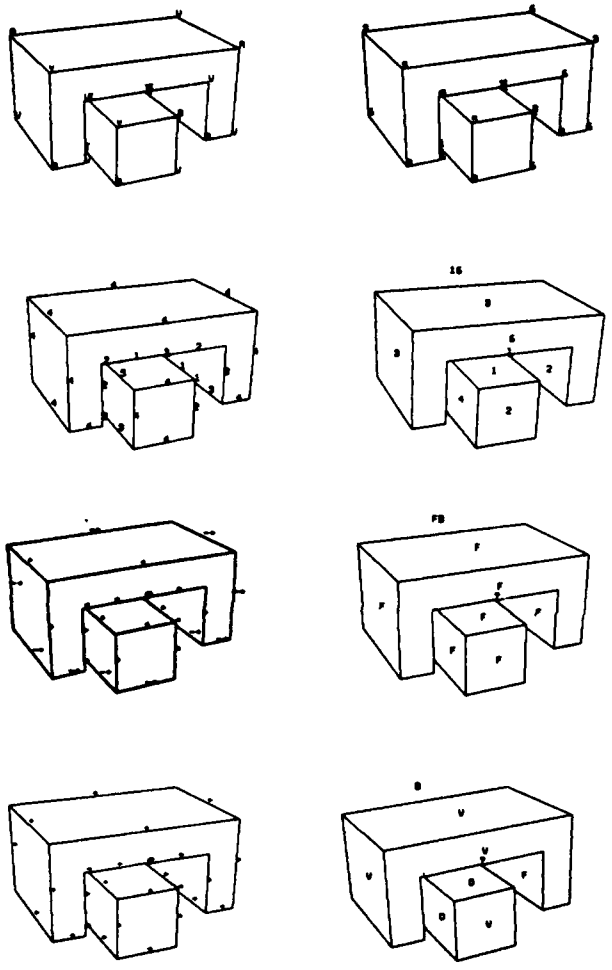


Figure 3.2. Parallel labeling of the blocks.

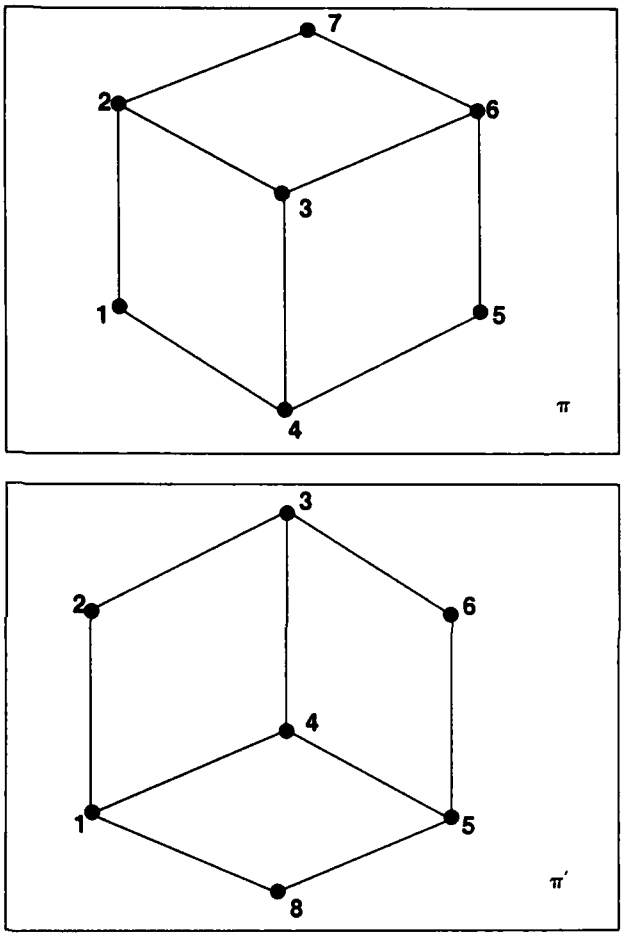


Figure 5.1. Two projections of a cube.

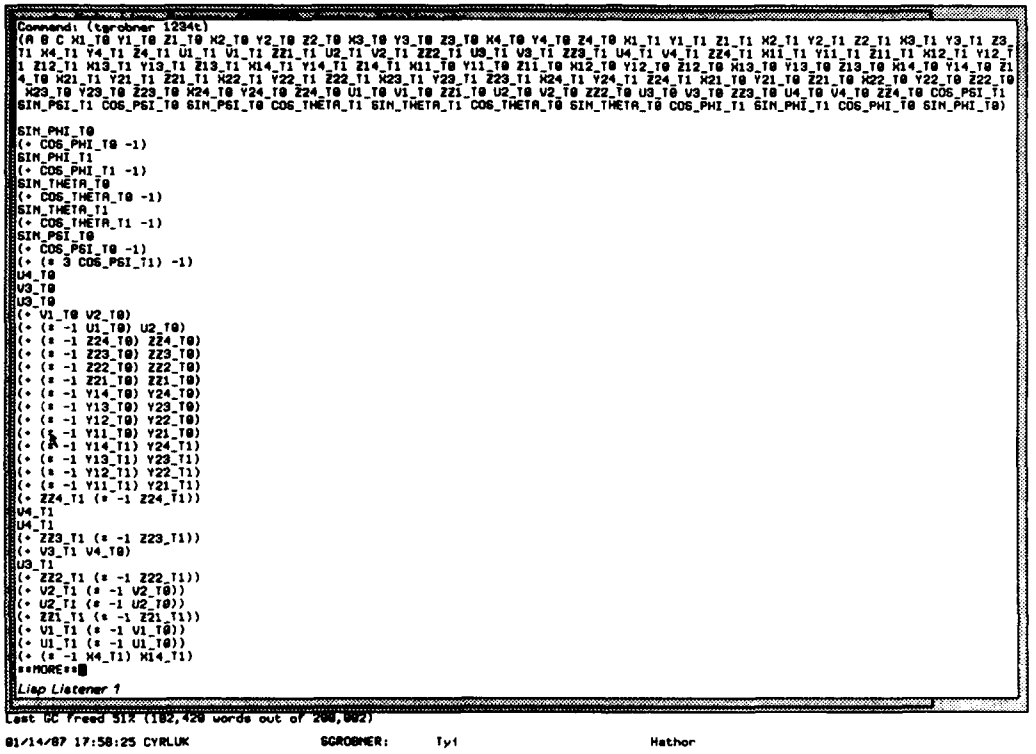


Figure 5.2. Part of a Gröbner basis input.

SPECTRAL AND POLARIZATION STEREO METHODS USING A SINGLE LIGHT SOURCE

Lawrence Brill Wolff

Department of Computer Science
Columbia University
New York, NY 10027

ABSTRACT

Proposed herein are novel stereo techniques for the determination of surface orientation from a single view and a single light source exploiting the physical wave properties of light and the basic physics of material surfaces. Conventional photometric stereo techniques consider the intersection of equi-reflectance curves that arise in gradient space while varying the imaging geometry with respect to the spatial position of a light source. Spectral and polarization stereo methods consider the intersection of equi-reflectance curves in gradient space while varying the wavelength (i.e. color) and/or the polarization of light emanating from a single light source while leaving the imaging geometry invariant. The reflectance function used here results from an extended form of the Torrance-Sparrow model which is dependent upon the wavelength and polarization of incident light as well as various physical parameters of the material surface. This reflectance model is more accurate than commonly used Lambertian reflectance models and is applicable to a wide variety of isotropically rough surfaces ranging from metals to paper. Computer simulations of the intersections of equi-reflectance curves that arise by varying incident wavelength and polarization are shown for two different types of materials; a dielectric, Magnesium oxide (used in white paint) and a conductor, Aluminum. Two different methods called direct and indirect resolution of specular and diffuse reflection components are used to minimize measurement error and to resolve intersections, respectively.

An interesting problem arises for stereo methods that vary only wavelength and/or polarization of the incident light source. Measurement of surface orientations that do not lie on the source-viewing axis line in gradient space determined by the origin and the point representing the incident orientation of the single light source cannot be uniquely resolved any better than up to a two point ambiguity. This results from an inherent "flip" symmetry induced on each equi-reflectance curve in gradient space by the isotropic reflectance function being used. It is demonstrated that it is feasible to break this inherent symmetry by performing spectral/polarization stereo methods using a single extended light source with a variable "asymmetric" aperature. This makes it possible to obtain a unique measurement for surface orientations not on the source-viewing axis using only a single light source.

1. INTRODUCTION

Previous stereo methods such as the traditional stereo method of depth determination by parallax and the photometric stereo method for the determination of local surface orientation ([Woodham 1980]) entail taking successive images while varying some aspect of the imaging geometry. Recently it was reported in [Wolff 1986] that stereo vision can be generally formalized far beyond previously used methods by taking successive images while varying physical parameters of an imaging system other than just the imaging geometry. Spectral and polarization stereo methods used to determine surface orientation are a particular example of a class of stereo methods predicted by this general stereo model.

The stereo methods presented here involve using the familiar imaging setup as depicted in figure 1 with a single light source incident on a material surface. Standard gradient space will be used throughout to represent surface orientation. Spectral stereo entails varying the incident wavelength alone between successive images. Polarization stereo entails varying the incident polarization alone between succesive images. Spectral and polarization stereo can be combined by varying both the wavelength and polarization simultaneously between successive images. Spectral stereo can be implemented by using various narrow pass monochrome color filters and polarization stereo can be implemented using various polaroid filters. All other physical parameters governing the imaging system are to remain unchanged. As in conventional photometric stereo, local surface orientation is determined by measuring the image irradiance value at the pixel corresponding to the projected object point giving rise to an equi-reflectance curve in gradient space. The projection function of an object onto the image plane is assumed to be orthographic.

Because of its simplicity, the Lambertian reflectance function is used in numerous applications in low level vision. However the reflection of light off a material surface is a highly complex process dependent on the imaging geometry, the wave characteristics of the incident light radiation, the micro-geometry of the material surface (e.g. roughness) and the internal physics (e.g. electric resistivity) of the surface material itself. These aspects of the reflection process beyond the imaging geometry account for why the reflective properties of most material surfaces deviate from the Lambertian model.

The motivation for using the extended Torrance-Sparrow reflection model (ETSRM) is twofold. First, as just mentioned, the Lambertian reflectance model is too simplistic and is only applicable to a very narrow class of materials whereas the ETSRM is far more general. Second is that the Lambertian model is not dependent at all on the wavelength or polarization of incident light which would make the stereo methods considered here infeasible. While the ETSRM may be relatively new to the computer vision community, it is popularly used in computer graphics for rendering realistic images of smooth material surfaces [Cook, Torrance 1981].

To demonstrate the wide applicability of the ETSRM, and therefore to spectral/polarization stereo methods, simulations of the measurement of surface orientation will be performed on two very different types of materials. The first is Magnesium oxide (MgO) classified as a *dielectric* which is a material that does not conduct electricity. MgO has a strong diffuse component of reflection. The second material is Aluminum which is a *conductor* and has a strong specular component of reflection. As will be seen, the distinction between whether a material is a dielectric or a conductor is important for selecting the appropriate spectral/polarization stereo technique.

The reflectance function resulting from the ETSRM is isotropic in the sense that the reflected radiance from a point on a material surface is independent of any rotation about the normal to this point. This induces a "flip" symmetry on the resulting equi-reflectance curves in gradient space. A formal description of this is given in section 7. Because the imaging geometry is left invariant for spectral/polarization stereo methods the equi-reflectance curves produced from successive images possess the same exact "flip" symmetry. This makes it theoretically impossible to measure most surface orientations uniquely beyond a two point ambiguity using spectral/polarization stereo methods with a single point light source. A technique is presented to break this "flip" symmetry by using a single extended light source subtending a solid angular area which does not possess the same "flip" symmetry.

Unless stated otherwise, all light sources mentioned are assumed to be point light sources. For brevity, the term *source-viewing plane* in this paper refers to the plane determined by the incident source orientation vector (represented by $(P_s, Q_s, -1)$) and the viewing orientation vector (assumed to be (0, 0, -1)). The *source-viewing axis* is the line in gradient space determined by the origin and the point (P_s, Q_s).

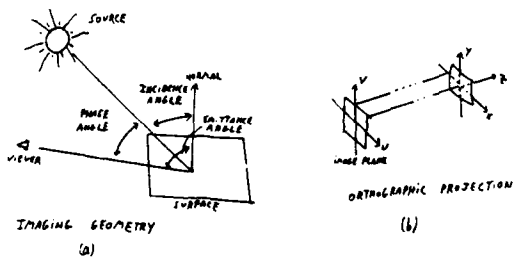


Figure 1

2. THE EXTENDED TORRANCE-SPARROW REFLECTION MODEL

The reflectance model reported in [Torrance, Sparrow 1967] assumes that every surface has a microscopic level of detail which consists of a statistically large number of perfectly smooth planar microfacets. These microfacets are oriented according to a particular probability distribution function of the angle between the normal to each microfacet and the normal to the surface. This implies that the surface is isotropically rough about the normal to the surface as the probability distribution function is not dependent on the azimuthal orientation of the normal to a microfacet.

The Torrance-Sparrow reflectance model is primarily based on geometric optics. Light rays are assumed to reflect off a material surface with both a specular and a diffuse component. The specular component of reflection accounts for when a light ray specularly reflects off a microfacet and the diffuse component is described by the Lambertian reflectance function which accounts for multiple specular reflections and/or reflections of light rays that penetrate into the skin of the surface and then reflect back out. According to the Torrance-Sparrow reflection model the form of the reflected radiance function is given by:

$$(1)\quad dN_r \equiv gN_iR_s d\omega_i + N_iR_d d\omega_i \ .$$

The terms R_s and R_d are the functions for the specular and diffuse components of reflection respectively. The term N_i represents the incident radiance of the light source through the infinitesimal solid angle $d\omega_i$ (see figure 2) and g is the proportion of the specular reflectance relative to the diffuse reflectance. The equivalence symbol in equation (1) and all other equations in this paper represents proportionality. The specular and diffuse component reflection functions are given by:

$$R_s = \frac{F(\Psi',\eta)\,G(\Psi,\Theta,\Phi)}{cos\Theta}P(\alpha)$$

$$R_d = cos\Psi$$

where the angular arguments correspond to figure 2. Note that R_d is the Lambertian reflectance function. Because of its central role in spectral/polarization stereo applications, section 3 is entirely devoted to the function $F(\Psi',\eta)$.

The function $P(\alpha)$ is the probability distribution function for the orientations of the normals to the planar microfacets. The probability distribution function proposed in [Torrance, Sparrow 1967] is:

$$P(\alpha) \equiv exp(-(\alpha c)^2 \ .$$

The variable α is the angle between the normal to a given microfacet and the normal to the surface and is depicted in figure 2. The term c is an ad hoc constant which is determined empirically by solving equation (1) using observed reflected radiance values.

It is suggested in [Cook, Torrance 1981] that a probability distribution function for the orientation of microfacet normals using physical parameters (i.e. no ad hoc constants) could be used to emulate the Beckmann distribution for mean scattered power of light reflected from a rough surface. This function is derived in [Beckmann, Spizzichino 1963] and is given by:

$$P(\alpha) \equiv \frac{1}{m^2 cos^4\alpha} exp(-\frac{tan^2\alpha}{m^2}) \ .$$

The term m is the root mean square slope value of the microfacet surface normals. Large values of m indicate a "rough" surface while small values of m indicate "smooth" surfaces. The extended Torrance-Sparrow reflection model refers to incorporating the Beckmann disribution function into the expression for the specular component of reflection R_s. The value for m can be determined by solving equation (1) using reflected radiance values, or can be measured more directly using a *stylus profilometer*.

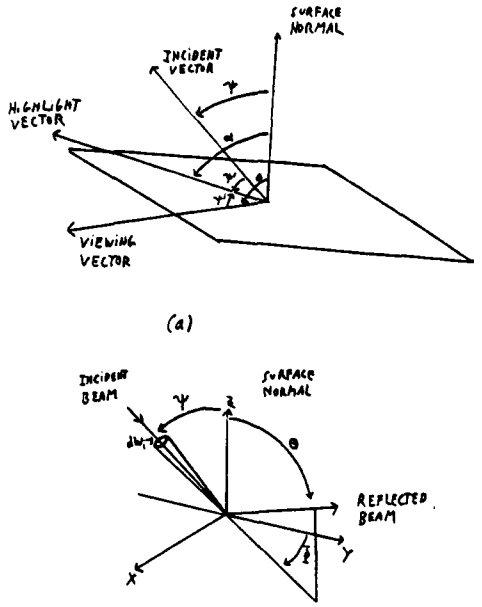


Figure 2

The reflection model presented in [Torrance, Sparrow 1967] also takes into account the mutual shadowing and masking of surface microfacets against one another. It is assumed that each specularly reflecting microfacet comprises one side of a symmetric V groove as depicted in figure 3. The process of shadowing and masking attenuates the total fraction of reflecting surface area. The proportional amount of this attenuation is called the *geometric attenuation factor* and is given by:

$$G(\Psi,\Theta,\Phi) = min\{\,1, \frac{2cos\alpha cos\Theta}{cos\Psi'}, \frac{2cos\alpha cos\Psi}{cos\Psi'}\,\} \ .$$

The above expressions for the geometric attenuation factor and the Beckmann distribution function conform to the geometry depicted in figure 2. With respect to the angles Ψ, Θ and Φ depicted in figure 2, α and Ψ' can be expressed as:

$$\Psi' = \frac{1}{2} cos^{-1}[\,cos\Theta cos\Psi - sin\Theta sin\Psi cos\Phi\,]$$
$$\alpha = cos^{-1}[\,cos\Psi cos\Psi' + sin\Psi sin\Psi' cos\beta\,]$$

where

$$\beta = sin^{-1}[sin\Phi sin\Theta / sin2\Psi'] \ .$$

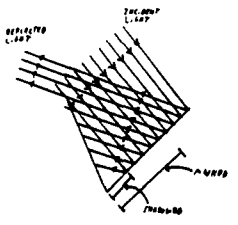


Figure 3

3. THE FRESNEL REFLECTION COEFFICIENT

The function $F(\Psi',\eta)$ incorporated into the specular component of reflection is of primary importance to spectral and polarization stereo. It is known as the *Fresnel reflection coefficient.* This is the only part of expression (1) that is dependent on the wavelength and polarization of incident light. The derivation of $F(\Psi',\eta)$ treats light as an electromagnetic wave. The polarization of a light wave is defined in terms of how the electric field vector is oriented with respect to the *plane of incidence*, the plane determined by the incident and specularly reflected vectors. An incident light wave is parallel polarized if its electric field vector is parallel to the plane of incidence and is perpendicularly polarized if its electric field vector is perpendicular to the plane of incidence.

An incident light wave that is specularly reflected from a perfectly smooth surface results in a partially transmitted wave and a partially reflected wave. The Fresnel reflection coefficient represents the proportional attenuation of the incident light energy per unit time per unit area for specular reflection off of a perfectly smooth surface. This simulates the specularly reflected radiance of light rays off of the perfectly smooth microfacets.

The angle Ψ' is the *specular angle of incidence* of a light ray upon microfacets with surface normals oriented parallel to the highlight vector depicted in figure 2. The specular angle of incidence on the microfacets of a surface is equal to 1/2 of the phase angle and is not dependent on the orientation of the surface. The number of microfacets that the incident light is specularly reflected from is determined by the surface orientation and the probability distribution function $P(\alpha)$.

The term $\eta = n - i\kappa$ is the *complex index of refraction* for the surface material. The real terms n and κ are dependent upon the wavelength of incident light λ_0 according to the formulas:

$$n^2 = \frac{\upsilon\gamma c_0^{\,2}}{2}\{\,1 + [\,1 + (\frac{\lambda_0}{2\pi c_0 r_e \gamma})^2\,]^{1/2}\,\}$$

$$\kappa^2 = \frac{\upsilon\gamma c_0^{\,2}}{2}\{\,-1 + [\,1 + (\frac{\lambda_0}{2\pi c_0 r_e \gamma})^2\,]^{1/2}\,\} \ .$$

The term c_0 is the speed of light in a vacuum, r_e is the electrical resistivity of the surface material, and υ and γ are the electrical permitivity and the magnetic permeability of the surface material respectively. It turns out that r_e, υ and γ are dependent upon the surface temperature and the external presence of electric and magnetic fields. This suggests other stereo methods to determine surface orientation. For instance, vary the temperature between each successive image, or for a conducting material vary the strength of an electrical current passing through it between successive images. Varying the strength of magnetic fields can also be used.

The term κ is called the *coefficient of extinction.* The coefficient of extinction is zero for dielectrics (because r_e is infinite) and is non-zero for conductors since they have finite electrical resistivity r_e. For dielectrics, the term n is the *simple index of refraction.*

A detailed derivation of the Fresnel reflection coefficient is given in [Siegel, Howell 1981]. It is represented as the linear superposition of the Fresnel reflection coefficients for perpendicular and parallel polarized incident light waves respectively. That is:

$$F(\Psi',\eta) = sF_{perp}(\Psi',\eta) + tF_{para}(\Psi',\eta) \quad s,t \geq 0 \;\; s+t=1 .$$

In turn

$$F_{perp}(\Psi',\eta) = \frac{a^2 + b^2 - 2a\cos\Psi' + \cos^2\Psi'}{a^2 + b^2 + 2a\cos\Psi' + \cos^2\Psi'}$$

$$F_{para}(\Psi',\eta) = \frac{a^2 + b^2 - 2a\sin\Psi'\tan\Psi' + \sin^2\Psi'\tan^2\Psi'}{a^2 + b^2 + 2a\sin\Psi'\tan\Psi' + \sin^2\Psi'\tan^2\Psi'} F_{perp}(\Psi',\eta)$$

where

$$2a^2 = [(n^2 - k^2 - \sin^2\Psi')^2 + 4n^2k^2]^{1/2} + n^2 - k^2 - \sin^2\Psi'$$

$$2b^2 = [(n^2 - k^2 - \sin^2\Psi')^2 + 4n^2k^2]^{1/2} - (n^2 - k^2 - \sin^2\Psi')$$

Unpolarized light is the equi-superposition of parallel and perpendicular states. Therefore the Fresnel reflection coefficient for unpolarized light uses s=t=1/2.

Simulations of the determination of local surface orientation will be performed using the reflectance maps predicted by the ETSRM for MgO and Aluminum. The accuracy of the original Torrance-Sparrow reflection model was tested on these two materials in [Torrance, Sparrow 1967] with good results. The empirical values for g, the relative proportion of specular to diffuse component reflected radiance, were also obtained for MgO and Aluminum in this paper (g_{MgO}=2, g_{Al}=7/8) and will be used in the simulations in future sections. The Fresnel reflection coefficients for MgO and Aluminum are graphed in figures 4 (a) thru (d) against the angle of incidence for different incident light polarizations and wavelengths. Be aware of the scale on the vertical axis. The values for the indices of refractions were obtained from [Physics Handbook] for different wavelengths.

Typical human vision perceives light waves with a wavelength between 400 and 700 nanometers (NM). "Vision" can theoretically exist for any wavelength of electromagnetic radiation. The wavelengths shown in figures 4 (a) and (b) for MgO are not in the visible part of the spectrum. These wavelengths were chosen to fall

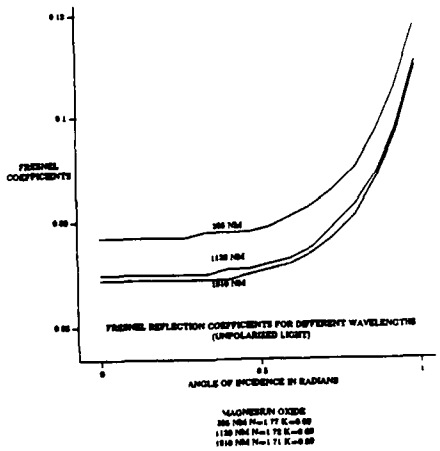


Figure 4(a)

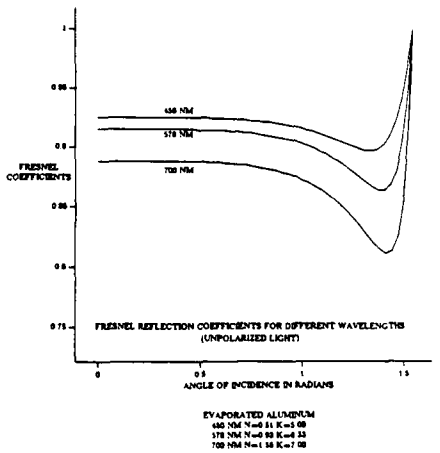


Figure 4(c)

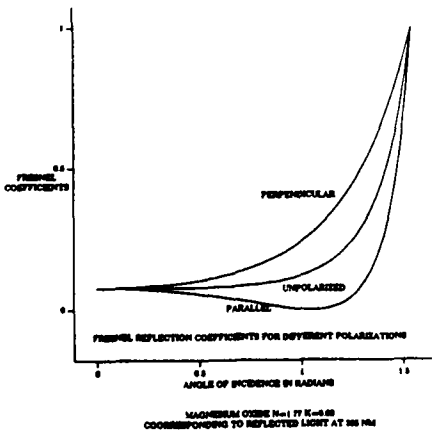


Figure 4(b)

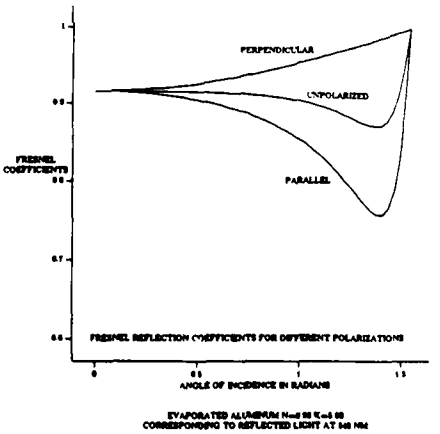


Figure 4(d)

as close to the visible part of the spectrum as possible while being able to obtain the indices of refraction from the tables in [Physics Handbook]. Interpolation of data was not used to make the simulation of equi-reflectance curves as realistic as possible.

An interesting feature of the dielectric MgO is that it has a *Brewster angle* at arctan(n=1.77) = 60.5° for which no parallel polarized light is reflected. All dielectrics have a Brewster angle at the arctangent of their simple index of refraction. For the determination of surface orientation by varying polarization this is a very important angle to know in a controlled environment. Accuracy of measurement can be substantially enhanced by making the phase angle approximately equal to twice the Brewster angle for a dielectric material. This will be seen in section 5.

As can be seen the simple index of refraction for MgO has a rather weak dependence on wavelength. This suggests that the combined specular and diffuse reflectance will be altered only slightly for different incident wavelengths of light. In section 6 a technique will be presented to obtain a discernable intersection from specular and diffuse component equi-reflectance curves.

4. ERROR ANALYSIS FOR THE INTERSECTION OF REFLECTANCE CURVES

Implementation of spectral/polarization stereo methods involve the physical measurement of image irradiance values which have associated with them an inherent error. This in turn generates an *error collar* about each equi-reflectance curve. Given an image irradiance measurement value I and a reflectance function R(p,q), the image irradiance equation gives:

$$I = R(p,q)$$

where then

$$\delta I = \frac{\partial R}{\partial p}\delta p + \frac{\partial R}{\partial q}\delta q \ .$$

Therefore, for a given inherent error of measurement in I, the width of the error collar about a point on the equi-reflectance curve resulting from the above image irradiance equation will decrease as the gradient of the reflectance function R(p,q) increases at this point.

Figure 5(a) depicts a nearly perpendicular intersection of two equi-reflectance curves along with their associated error collars. The curved quadrilateral region which is cross hatched represents the area in which the true surface orientation lies. Without referring to any particular probability distribution function defined on this cross hatched area, the *expected error* of measurement of surface orientation derived from this intersection can be qualitatively interpreted as the area of the cross hatched region while the *worst case error* can be qualitatively interpreted as the maximum distance between two points in the cross hatched region.

Figure 5(b) depicts a more oblique intersection of two equi-reflectance curves along with their associated error collars. Assuming that the gradients of the reflectance functions within the cross hatched regions are roughly equal between figures 5 (a) and (b), an oblique intersection increases both the expected and the worst case errors for the measurement of surface orientation from the intersection of equi-reflectance curves. Sometimes an oblique intersection may occurr in a region of gradient space where the

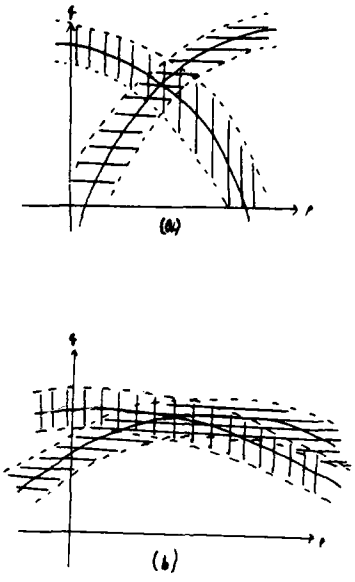


Figure 5

gradient of the reflectance function is high making both the expected and worse case errors less than for the errors associated with a perpendicular intersection of equi-reflectance curves in a region of gradient space where the gradient of the reflectance function is quite low.

Optimal criteria for the overall minimization of expected and worst case errors are extremely complex and are dependent on a multitude of assumptions including *a priori* knowledge about the expected range for the surface orientation to be measured. Simultaneous maximization of perpendicularity of intersections and maximization of the gradient of the reflectance function over a global set of gradient space points is impossible in most cases.

A good strategy to increase the gradient of the reflectance function is to increase the phase angle. This involves a tradeoff as increasing the phase angle increases the area of gradient space that lies in shadow. A good strategy for optimizing perpendicular intersections for spectral/polarization methods using the ETSRM is to pick incident wavelengths and polarizations that will produce intersections of specular and diffuse component equi-reflectance curves. This is indicated in figure 6 which shows an overlay of the specular and diffuse component reflection functions for MgO. Near perpendicular intersections occurr almost everywhere in the first quadrant for a light source with incident orientation (7.0, 3.0, -1). The same overlay of specular and diffuse reflectance maps would result for Aluminum except that reflected radiance values corresponding to each equi-reflectance contour would be different than from the ones in figure 6.

It turns out that *direct resolution* of specular and diffuse components for any dielectric is possible using polarization stereo for selected phase angles close to twice the Brewster angle. This is described in section 5. However, as reflection of light from dielectrics is very insensitive to variations in wavelength, the intersection of equi-reflection curves for even vast shifts in wavelength are so oblique that the equi-reflectance curves themselves are almost indistinguishable. Even though reflection of

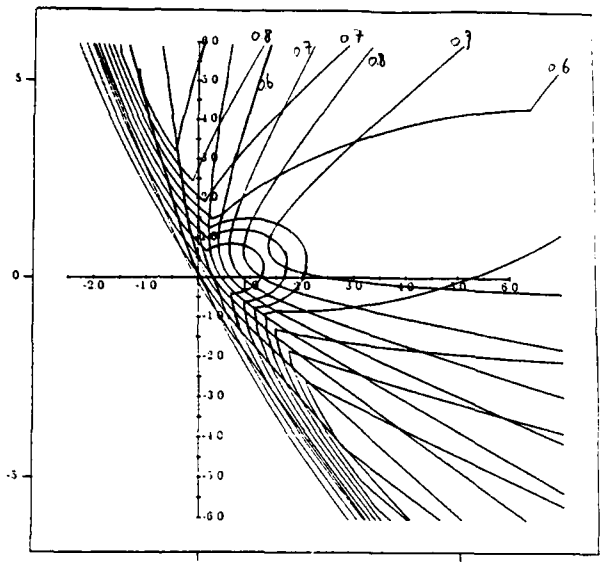


Figure 6

light from conductors has a stronger dependence on wavelength and a moderate dependence on polarization (see figures 4 (c) and (d)), this problem also occurrs for spectral/polarization stereo methods applied to conductors. These problems are so bad that minimization of error is surmounted by the problem of ascertaining an intersection in the first place. It is required therefore to use a methodology described in section 6 called *indirect resolution* of specular and diffuse reflection components.

5. POLARIZATION STEREO FOR DIELECTRICS

It was mentioned in section 3 that for all dielectrics there exists an angle of incidence, called the Brewster angle, at which parallel polarized light is not reflected. That is, the Fresnel reflection coefficient for parallel polarized light at the Brewster angle is zero. Figure 4(b) shows that in fact the Fresnel reflection coefficient for parallel polarized light is small over a large range of angles of incidence less than the Brewster angle. By observing the expression for R_s in section 2, this phenomenon can be exploited to "scale" the specular component of reflection to nearly zero thus isolating the diffuse component of reflection R_d.

Suppose that two image irradiance measurements are taken for parallel polarized and perpendicular polarized incident light respectively and that the phase angle is equal to exactly twice the Brewster angle for a given dielectric. Therefore the specular angle of incidence of light on the microfacets will be exactly the Brewster angle itself. The first equi-reflectance curve resulting from the incident parallel polarized light will be exactly the same as for the Lambertian model at this angle of incidence. Since a moderate amount of perpendicular polarized light is reflected at the Brewster angle for a dielectric, the second equi-reflectance curve will have a significant specular component of reflection. Even though the second equi-reflectance curve is not a pure specular component of reflection, it will tend to be a curve that will have a strong intersection with the Lambertian reflectance curve. This technique of forcing the total reflectance curves significantly towards the specular and diffuse component reflection functions respectively is what is termed *direct resolution* of specular and diffuse components.

An example of this technique is shown in figure 7 for MgO. Unfortunately one disadvantage is that the Brewster angle for MgO is 60.5° which would make the phase angle equal to about 121°. This would leave much of gradient space in shadow. Figures 11 (a) through (c) use an incident source orientation vector of (7.0, 3.0, -1) which gives a phase angle of approximately 82.6°. The specular angle of incidence is therefore about 41.3°, nearly 20° from the Brewster angle for MgO. Still, strong intersections are noted in the overlay of the parallel polarized and the perpendicularly polarized reflectance maps. This is due to the fact that the Fresnel reflection coefficient for parallel polarized light is still negligable at this angle of incidence and that the Fresnel reflection coefficient for perpendicularly polarized light is approximately 4 times that for parallel polarized light (see figure 4(b), 41.3° is about 0.720 radians).

Figure 8 depicts an actual simulation of the determination of local surface orientation from polarization stereo for a point on a sphere with a Magnesium oxide surface. The sphere is assumed to have radius 25 and the surface orientation is to be determined for the point on the sphere corresponding to the image point (12,18). The equation for the visible part of a sphere of radius r using the coordinate system depicted in figure 1.b is:

$$z = -\sqrt{r^2 - x^2 - y^2}$$

which implies that

$$\frac{\partial z}{\partial x} = \frac{x}{\sqrt{r^2 - x^2 - y^2}}, \quad \frac{\partial z}{\partial y} = \frac{y}{\sqrt{r^2 - x^2 - y^2}}.$$

Therefore, the local surface orientation at the point corresponding to the image coordinate (12,18), assuming orthographic projection, is approximately (0.958, 1.437, -1) in gradient space representation.

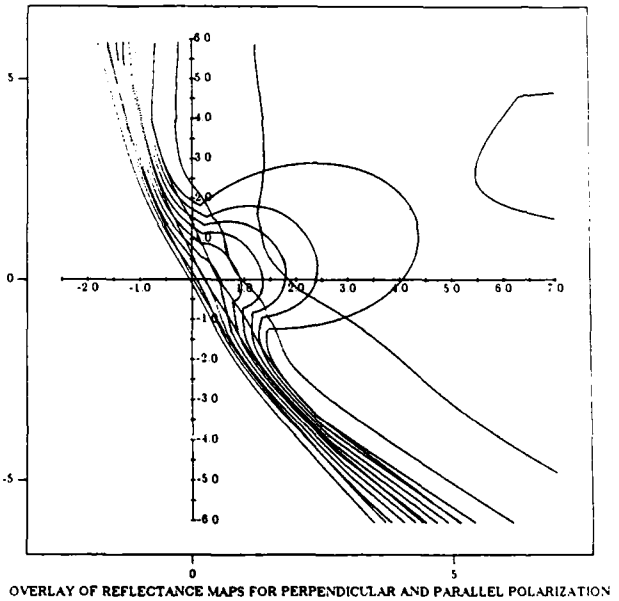


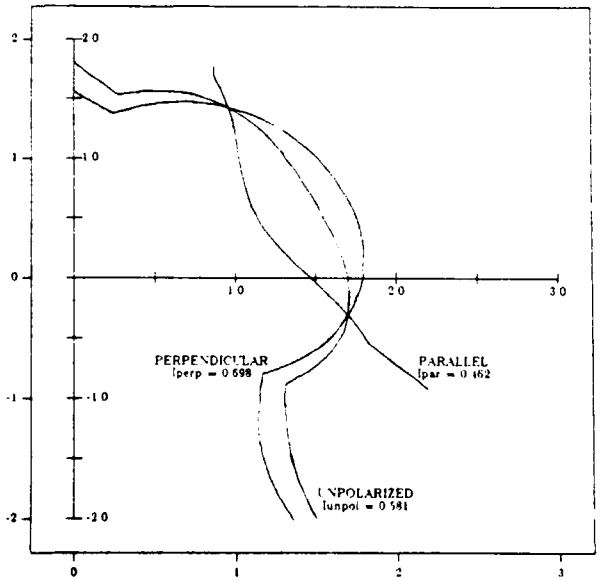
Figure 7

The equi-reflectance curves in figure 8 result from three different incident polarizations respectively. It is clear that an intersection occurrs at the correct orientation point. This intersection could be obtained by using the two equi-reflectance curves for parallel polarization and perpendicular polarization respectively. However there is an additional intersection point that could just as well be the true value of the surface orientation. The third equi-reflectance curve produced from unpolarized incident light does not provide more information with respect to resolving between these two intersection points. In fact, for this same configuration of the imaging system, all equi-reflectance curves produced from any incident polarization or wavelength will go through these same two points in gradient space. This is due to an inherent "flip" symmetry about the source-viewing axis that cannot be broken by varying the value of the Fresnel reflection coefficient, $F(\Psi',\eta)$.

It is possible to break this "flip" symmetry by using a second light source which has an incident orientation not lying on the source-viewing axis of the original light source. This would produce an equi-reflectance curve going through the true surface orientation point and not going through the second intersection point observed for polarization stereo. However, the "flip" symmetry inherent to spectral/polarization stereo can also be broken by using a single extended light source with a variable aperature. This will be discussed in section 7.

6. SPECTRAL AND POLARIZATION STEREO BY INDIRECT RESOLUTION INTO SPECULAR AND DIFFUSE REFLECTION COMPONENTS

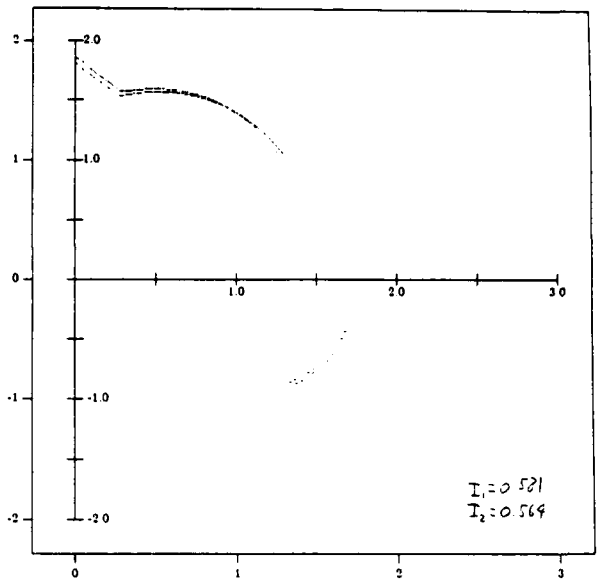
The equi-reflectance curves in figures 9 and 10 attempt to measure the same surface orientation in the problem that is described in section 5. For figure 10, the surface of the sphere is assumed to be made of Aluminum. By observing figures 9 and 10 it is evident that directly obtaining an intersection point for reflectance curves for spectral stereo applied to dielectrics, or for any combination of



INTERSECTION OF EQUI-REFLECTANCE CURVES FOR DIFFERENT POLARIZATIONS

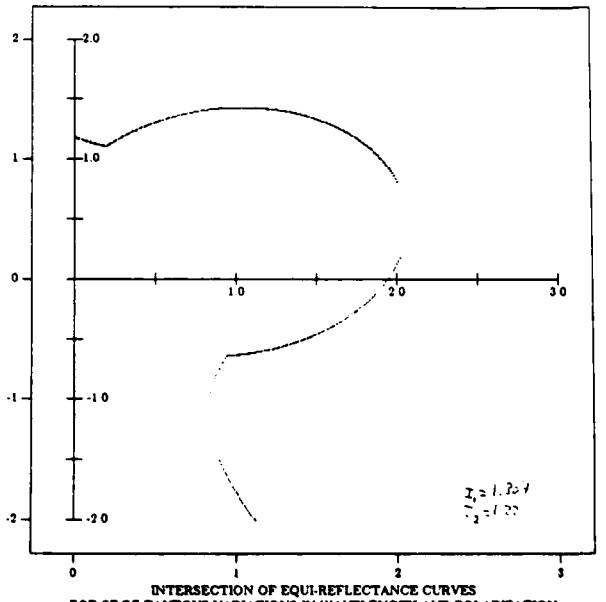
COMBINED SPECULAR AND DIFFUSE REFLECTANCE MAP
REFLECTANCE FUNCTION 0 5*(2 0*Rs + Rd)
SOURCE VECTOR ORIENTATION Ps=7 0 Qs=3 0
MAGNESIUM OXIDEN=1 77 K=0 00 (365 NM)
RMS SLOPE VALUE= 2 0

Figure 8



INTERSECTION OF EQUI-REFLECTANCE CURVES FOR TWO DIFFERENT WAVELENGTHS
COMBINED SPECULAR AND DIFFUSE REFLECTANCE MAP
REFLECTANCE FUNCTION 0 5*(2 0*Rs + Rd)
SOURCE VECTOR ORIENTATION Ps=7 0 Qs=3 0
MAGNESIUM OXIDE N=1 77 K=0 00 (365 NM) AND N=1 72 K=0 0 (1130 NM)
RMS SLOPE VALUE= 2.0

Figure 9



INTERSECTION OF EQUI-REFLECTANCE CURVES
FOR SIMULTANEOUS VARIATIONS IN WAVELENGTH AND POLARIZATION

COMBINED SPECULAR AND DIFFUSE REFLECTANCE MAP
REFLECTANCE FUNCTION: 0 5*(0 875*Rs + Rd)
SOURCE VECTOR ORIENTATION Ps=7 0 Qs=3.0
ALUMINUM
N=0 51 K=5.00 (450 NM) PERPENDICULAR POLARIZATION
N=1 55 K=7.00 (700 NM) PARALLEL POLARIZATION
RMS SLOPE VALUE= 2.0

Figure 10

spectral and polarization stereo applied to conductors, is very difficult. Spectral stereo applied to dielectrics is hard to implement because the simple index of refraction is very insensitive to changes in wavelength. Unless the sampling rate for the equi-reflectance curves in gradient space is very high, two curves obtained for different incident wavelengths will appear indistinguishable. Another problem for spectral stereo applied to dielectrics is the necessity for low measurement error in the image irradiance. The shift in the observed reflected radiance between the two equi-reflectance curves in figure 9 is approximately 3.0% .

The observed image irradiance shift for the two equi-reflectance curves in figure 10 is fairly significant being almost 9% . The problem of indistinguishability between equi-reflectance curves here is even worse than in figure 9. This is due to the relative invariance of the shape of the equi-reflectance curves for Aluminum for fairly significant changes in the reflected radiance.

A technique is presented here that will use two reflected radiance measurements to solve for the values of the specular and diffuse components of reflection respectively from two simultaneous equations. The functions R_s and R_d are the same as those defined in section 2 except that the R_s used here is gR_s. Define r_s such that $R_s=F(\Psi',\eta)r_s$. Suppose that two reflectance measurements I_{ref}^1 and I_{ref}^2 are taken from two sucessive images at the same pixel. Also, the value of the Fresnel term with respect to this pixel for the first and second image are $F^1(\Psi',\eta)$ and $F^2(\Psi',\eta)$ respectively. Then the following equations arise:

$$F^1(\Psi',\eta)r_s + R_d = I_{ref}^1 \pm \varepsilon$$

$$F^2(\Psi',\eta)r_s + R_d = I_{ref}^2 \pm \varepsilon$$

where ε is the error inherent in the reflectance measurement. The solutions for R_s and R_d are:

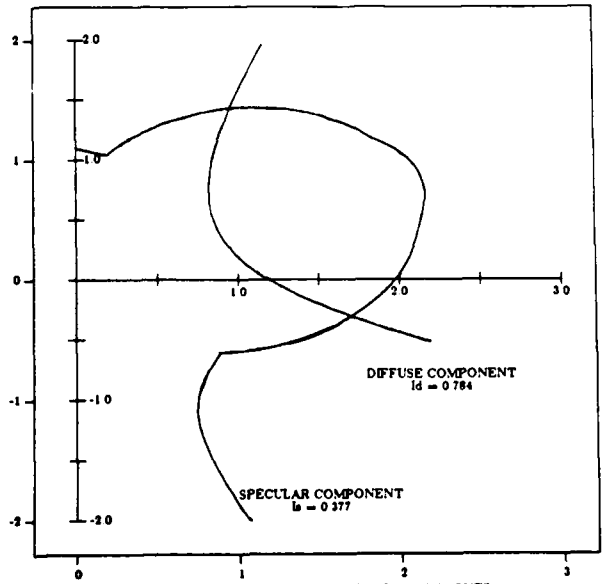
$$R_s = F^1(\Psi',\eta)\frac{I_{ref}^1 - I_{ref}^2}{F^1(\Psi',\eta) - F^2(\Psi',\eta)} \pm 2\frac{F^1(\Psi',\eta)}{F^1(\Psi',\eta) - F^2(\Psi',\eta)}\varepsilon$$

$$R_d = \frac{F^1(\Psi',\eta)I_{ref}^2 - F^2(\Psi',\eta)I_{ref}^1}{F^1(\Psi',\eta) - F^2(\Psi',\eta)} \pm \frac{F^1(\Psi',\eta) + F^2(\Psi',\eta)}{F^1(\Psi',\eta) - F^2(\Psi',\eta)}\varepsilon .$$

Note that the solution for R_s is for the specular component of the total reflectance measured in the first image. This technique of decomposing the value of a total reflectance function into its specular and diffuse component values is what is termed *indirect resolution* of specular and diffuse components.

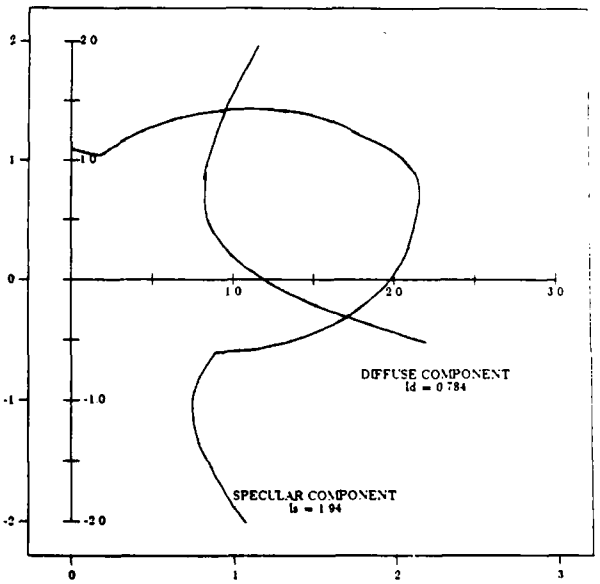
Thus, even for small shifts in the Fresnel reflection coefficient, two measurements of the reflected radiance make it possible to obtain the specular and diffuse component reflection values for each measurement. These values in turn can be used to generate an equi-reflectance curve for the specular and diffuse component respectively which produces strongly detectable intersections as mentioned in section 4. While the detection of intersection problem may be solved, the associated error collars may be huge if the experimental measurement error ε is significant and the difference between the first and second Fresnel reflection coefficients is very small. This can be seen from the equations immediately above.

Figures 11 and 12 show the intersection of specular and diffuse component equi-reflectance curves obtained from the same reflectance measurements that generated the curves in figures 9 and 10, respectively. Of course the experimental error ε for this



RESOLVED SPECULAR AND DIFFUSE COMPONENTS
SOURCE VECTOR ORIENTATION Ps=7.0 Qs=3.0
MAGNESIUM OXIDE N=1.77 K=0.00 (365 NM)
SECOND REFLECTANCE MEASUREMENT OBTAINED AT N=1.72 K=0.00 (1130 NM)
UNPOLARIZED LIGHT
RMS SLOPE VALUE= 2.0

Figure 11



RESOLVED SPECULAR AND DIFFUSE COMPONENTS
SOURCE VECTOR ORIENTATION Ps=7 0 Qs=3 0
ALUMINUM N=0 81 K=5 00 (450 NM) PERPENDICULAR POLARIZATION
SECOND REFLECTANCE MEASUREMENT OBTAINED AT N=1 55 K=7 00 (700 NM) PARALLEL POLARIZATION
RMS SLOPE VALUE= 2 0

Figure 12

simulation is assumed to be 0. The fact that the curves in figures 11 and 12 are exactly the same should be no surprise as equi-reflectance curves are invariant with respect to change in scale. The Fresnel reflection coefficient merely scales the specular component of reflection. Only the observed specular reflected radiance is different between the two figures.

The same "flip" symmetry that was discussed in section 5 is also inherent to the individual functions R_s and R_d. Additional reflectance measurements from different incident wavelengths and polarizations would not resolve the two point intersection ambiguity. To achieve unique measurement of surface orientation, the technique presented in section 7 using an extended light source will have to be implemented.

Spectral stereo for dielectrics for determining surface orientation is not generally recommended. Errors in the measurement of surface orientation for conductors using spectral/polarization methods can be reduced by simultaneously varying the wavelength and polarization as is done in figures 10 and 12. The shift in the Fresnel reflection coefficients from the first to the second image can be maximized by using parallel incident polarization at a high wavelength and then using perpendicular polarization at a low wavelength.

7. SYMMETRY BREAKING USING A SINGLE EXTENDED LIGHT SOURCE

The "flip" symmetry about the source-viewing axis that exists for the equi-reflectance curves produced by the ETSRM can most easily be shown by equating the gradient space representation (p, q, -1) with surface orientation represented in the spherical coordinate system depicted in figure 2 (b) with the z-axis being the viewer orientation. That is,

$$\frac{1}{\sqrt{1+p^2+q^2}}(p,q,-1) = (sin\Theta sin\Phi, sin\Theta cos\Phi, -cos\Theta) .$$

This implies that:

$$p = r sin\Phi$$

(2)

$$q = r cos\Phi$$

where:

$$r = \sqrt{p^2+q^2} = tan\Theta .$$

This demonstrates that the points (p,q) in gradient space can be identified with surface orientation using spherical coordinates via the *gnomonic projection* of the unit Gaussian sphere onto the tangent plane at the north pole. Points on the unit Gaussian sphere representing surface orientation are projected onto this plane by rays which emanate from the center of the sphere and go through the respective point.

The condition that the probability distribution of microfacets, $P(\alpha)$, be isotropic implies that there is a mirror symmetry with respect to the invariance of reflected radiance for surface orientation normals that differ azimuthally in Φ by the same angle about the source-viewing plane. The orientation component Θ is assumed to be the same for both normals. The reflected radiance is the same for both surface orientations because the parameters Ψ, Ψ', Θ and α are automatically the same. It is clear that expression (1) in section 2 will yield the same value under this equivalence. This results by observing that the specular and diffuse component reflected radiance are each left invariant.

The source-viewing axis in gradient space is the line at the constant angle $\tan^{-1}(P_s/Q_s)$ from the q-axis. The invariance of reflected radiance about the source-viewing plane described above implies that separate points that are equi-distant from the origin of gradient space and that make the same angle with the source-viewing axis must lie on the same equi-reflectance curve. This can be expressed succintly as the invariance of all equi-reflectance curves under the "flip" symmetry transformation:

$$\Phi \rightarrow 2tan^{-1}(P_s/Q_s) - \Phi$$

where the angle Φ is measured clockwise from the q-axis. Doing some algebra using equations (2) above, this transformation can be expressed in terms of gradient space variables as:

$$p \rightarrow \frac{P_s^2-Q_s^2}{P_s^2+Q_s^2}p + 2\frac{Q_sP_s}{P_s^2+Q_s^2}q$$

(3)

$$q \rightarrow 2\frac{Q_sP_s}{P_s^2+Q_s^2}p - \frac{P_s^2-Q_s^2}{P_s^2+Q_s^2}q .$$

Since the shift in the value of the Fresnel reflection coefficient does not alter the "flip" symmetry of the specular reflectance function during the process of spectral and/or polarization stereo using a single point light source, it is clear that a two point ambiguity will always exist for the measurement of surface orientations not on the source-viewing axis. From equations (3) it can be deduced that the same "flip" symmetry does not hold separately for a second light source with incident orientation $(P_s', Q_s', -1)$ if and only if (P_s, Q_s) and (P_s', Q_s') are linearly independent. That is, a second light source will have a different "flip" symmetry if and only if its incident orientation does not lie on the source-viewing axis of the first light source.

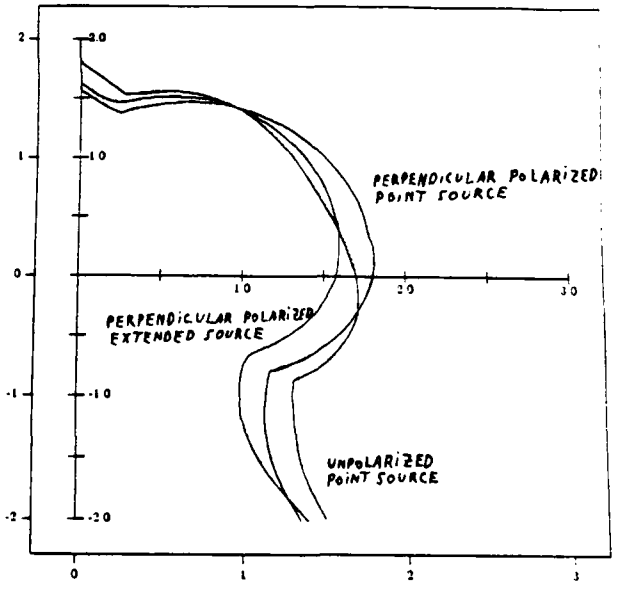
Suppose now that a second light source not lying on the source-viewing axis of the first light source was incident simultaneously with the first. That is the reflected radiance is now measured while both light sources are on. The "flip" symmetry of the first light source would effectively be broken for the resulting equi-reflectance curves. If at least a two point ambiguity existed in measuring surface orientation for all incident wavelengths and polarizations, the symmetry producing the ambiguity would not be the original "flip" symmetry. Such symmetries may occurr, but are highly unlikely for the measurement of most surface orientations.

Suppose that n point light sources are now simultaneously incident on a material surface. It can be shown using a simple geometric argument that the resulting equi-reflectance curves possess the "flip" symmetry of one of the n light sources being used if and only if the configuration of the incident orientation points in gradient space of the n light sources is in turn "flip" symmetric about the source-viewing axis of one of the n light sources. A single extended light source can be viewed as a continuous distribution of a large number of point light sources. Therefore the equi-reflectance curves

produced from a single extended source will possess a "flip" symmetry if and only if the area in gradient space subtended by the continuous distribution of incident orientation points is in turn "flip" symmetric.

Figure 13 shows the unique measurement of the surface orientation for the problem in section 5 for a MgO surface. Three reflected radiance measurements were taken altogether. Two of them used unpolarized and perpendicular polarized incident light from a single point source. The third reflected radiance measurement was taken for perpendicular incident light from an asymmetric "half-moon" extended source depicted in figure 14. The half angle used is 0.4 radians for the extended source and the expression for the reflected radiance in section 2 was numerically integrated over the solid angle subtended by the source. The size of the extended source was selected to be large to emphasize the process of breaking the inherent "flip" symmetry. Unique surface orientation measurements from much smaller extended sources are possible.

Of course all three reflected radiance measurements can be taken from the same light source with a variable aperature diaphram in front. For the first two measurements the aperature is a small "pinhole". For the third measurement the aperature is changed to the desired asymmetric shape and size. As mentioned above, it is possible for other more complicated symmetries to exist for resulting reflectance curves that may hinder the process of measuring certain surface orientations from spectral/polarization methods. Further changes to the shape of the extended source may then be useful, but at least the primary "flip" symmetry is broken.



COMBINED SPECULAR AND DIFFUSE REFLECTANCE MAP
R = 0.3*(2.0*Rs + Rd)
Ps=7.0 Qs=3.0
MAGNESIUM OXIDE N=1.77 K=0.00
RMS SLOPE VALUE= 2.0

Figure 13

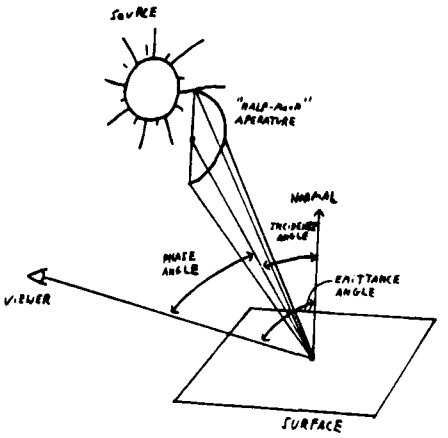


Figure 14

8. CONCLUSION

It has been demonstrated through computer simulation of equi-reflectance curves that arise from the extended Torrance-Sparrow reflection model that the measurement of surface orientation is obtainable by using stereo techniques that vary the wavelength and/or the polarization of incident light from a single light source while leaving the imaging geometry invariant. Different techniques were explored using two very different types of materials, Magnesium oxide and Aluminum. This was to show the applicability of spectral/polarization stereo methods to a large class of material surfaces.

The most viable technique presented was that of polarization stereo for dielectrics. This was based on the observation that the specular component equi-reflectance curves exhibit strong intersections with diffuse component equi-reflectance curves over a large region of gradient space. Because the Fresnel reflection coefficient for parallel polarized light is small in a large region of angles of incidence below the Brewster angle, it is possible to force the total reflectance function to be almost identical to its pure diffuse (i.e. Lambertian) reflection component for a phase angle well below 90°. The Fresnel reflection coefficient for perpendicular polarized light in this region is significant forcing the total reflectance function close to its specular reflection component. Thus using only two measurements of reflected radiance from parallel polarized and perpendicular polarized incident light respectively, surface orientation can be measured with relatively small error.

Spectral stereo for dielectrics is considered to be the least practical due to the insensitivity of the simple index of refraction to changes in wavelength. Spectral/polarization stereo for conductors is best when wavelength and polarization are simultaneously varied so as to maximize the difference in the Fresnel reflection coefficients between successive images. The main issue for spectral stereo applied to dielectrics and spectral/polarization methods applied to conductors is resolution of intersection points. In these cases the shapes of equi-reflectance curves are relatively invariant to changes in incident wavelength and polarization even though the change in the reflected radiance for conductors can be fairly significant. A technique was presented to resolve the specular and diffuse component values of the reflected radiance by solving two image irradiance equations simultaneously. Surface orientation is measured

from the strongly discernable intersection of specular and diffuse component equi-reflectance curves. However, for small variations in the Fresnel reflection coefficient, the associated errors can be quite high.

Finally it was formally shown how an isotropic reflectance function induces a "flip" symmetry on resulting equi-reflectance curves in gradient space about the source-viewing axis. Using a point light source it is therefore theoretically impossible to measure better than a up to a two point ambiguity, surface orientations that do not lie on the source-viewing axis using spectral/polarization techniques. A technique using a single extended light source was presented to resolve this ambiguity by breaking the inherent "flip" symmetry. By using an aperature which itself is not "flip" symmetric, equi-reflectance curves that pass through the true surface orientation point can be produced that do not pass through the false second point of ambiguity.

Experimentation is currently underway to empirically verify the theoretical development presented in this paper.

ACKNOWLEDGEMENTS

Many thanks go to the people that reviewed earlier drafts of this work. In particular, Terry Boult provided a very thoughtful review and helpful suggestions.

REFERENCES

[Beckmann and Spizzichino 1963]
Beckmann, P., and Spizzichino, A.
The Scattering of Electromagnetic Waves from Rough Surfaces
Macmillan, 1963

[Cook and Torrance 1981]
Cook, R.L., and Torrance, K.E.
A Reflectance Model For Computer Graphics
SIGGRAPH 1981 Proceedings, vol.15 #3 pp.307-316, 1981

[Physics Handbook]
American Institute of Physics Handbook
Third Edition
McGraw-Hill, 1972

[Siegel and Howell 1981]
Siegel, R. and Howell, J.R.
Thermal Radiation Heat Transfer
McGraw-Hill, 1981

[Torrance and Sparrow 1967]
Torrance, K.E., and Sparrow, E.M.
Theory for Off-Specular Reflection From Roughened Surfaces
Journal of The Optical Society of America, vol. 57 #9 pp.1105-1114, September 1967

[Wolff 1986]
Wolff, L.B.
Physical Stereo For Combined Specular and Diffuse Reflection
Tech report CS-242-86, Columbia University, Nov. 1, 1986

[Woodham 1980]
Woodham, R.J.
Photometric Method for Determining Surface Orientation from Multiple Images
Optical Engineering, vol.19 #1 pp.139-144, Jan-Feb 1980

---

[1]This research was supported in part by ARPA grant #N00039-84-C-0165.

SURFACE CURVATURE AND CONTOUR FROM PHOTOMETRIC STEREO

Lawrence Brill Wolff

Department of Computer Science
Columbia University
New York, NY 10027

ABSTRACT

Photometric stereo has been used to derive the normal map for a smooth surface depicting local surface orientation at each point. Many robotic vision applications require higher level descriptions of surface shape such as Gaussian curvature and contour lines. Presented here is an enhancement of conventional photometric stereo that directly computes the Gaussian curvature as well as principal curvatures and their directions at each point on an arbitrary smooth surface. First, image irradiance values from successive images are used to determine local surface orientation. Then image gradient values from the same set of successive images are used to compute the image Hessian matrix which gives complete knowledge about the second order rate of change of the smooth surface with respect to the image plane. It is shown that this provides enough information to compute the curvature matrix for a visible point on the smooth surface. The Gaussian curvature for each point is generated by taking the determinant of the curvature matrix. Principal directions of curvature are the eigenvectors of the curvature matrix. Lines of curvature can be generated by piecing together tangent lines that run parallel to a principal direction of curvature at each point.

1. INTRODUCTION

The photometric stereo technique presented here makes use of the *image Hessian* matrix presented in [Woodham 1981]. The image Hessian has also been referred to as the *curvature matrix* of the smooth surface being imaged. There is however a significant difference between the Hessian matrix for a smooth surface parameterized by its height above the image plane and the curvature matrix derived from the standard definition using the differential of the Gauss map. It will be shown in section 2 that while the determinant of the Hessian matrix agrees in sign with the Gaussian curvature that these two values are different quantitatively.

Computing the sign of the determinant of the Hessian matrix obtained from photometric stereo provides a quick method for determining whether the smooth surface is convex, concave or flat at a point. The curvature matrix at a point can be quantitatively derived from the components of the Hessian matrix and knowledge of the local surface orientation. Therefore Gaussian curvature and lines of curvature can be directly computed from image values without referring to neighboring values for normals from the normal map.

It was shown in [Woodham 1981] that the equations used to solve for the image Hessian matrix from a single image are underconstrained solving for three variables with two equations. Auxiliary constraints can aid in the process of deriving the image Hessian from a single image. For instance for developable surfaces (which results in at least one principle curvature being zero) all components of the image Hessian can be solved completely. The photometric stereo method presented here eliminates the need for any additional assumptions about the smooth surface with respect to fully recovering the image Hessian.

It is demonstrated in section 3 that once local surface orientation is known, the image Hessian can be completely solved for an arbitrary smooth surface from two or more successive images. Therefore the derivation of the image Hessian can be combined with conventional photometric stereo by first computing local surface orientation from successive images and then using at least two of these successive images to solve for the components of the image Hessian directly. Even though using more successive images would severely overconstrain the solution for the Hessian matrix, the solution would be more statistically accurate using experimental values.

2. THE IMAGE HESSIAN AND THE CURVATURE MATRIX

The image irradiance equation relating image irradiance I(x,y) to reflected scene radiance is written as:

$$I(x,y) = R(p,q)$$

where R(p,q) is the reflectance map as a function of gradient space variables p and q. If the height of the smooth object surface above the image plane is given by the twice *differentiable function* $h(x,y)$ then the smooth object surface can be parameterized using image coordinates by (x, y, h(x,y)) and

$$p = \frac{\partial h}{\partial x} \quad q = \frac{\partial h}{\partial y}.$$

The coordinate system used here is depicted in figure 1. Orthographic projection of the object surface onto the image plane is assumed throughout. Taking the partial derivative of the image irradiance equation with respect to the variables x and y gives the two equations

$$I_x = \partial p/\partial x R_p + \partial q/\partial x R_q$$
$$I_y = \partial p/\partial y R_p + \partial q/\partial y R_q$$

which in turn gives rise to the vector equation

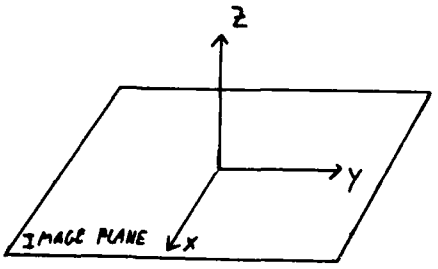


Figure 1

$$(1)\quad \begin{pmatrix} I_x \\ I_y \end{pmatrix} = \begin{pmatrix} \partial p/\partial x & \partial q/\partial x \\ \partial p/\partial y & \partial q/\partial y \end{pmatrix} \begin{pmatrix} R_p \\ R_q \end{pmatrix}.$$

The image Hessian is defined as

$$\begin{pmatrix} \partial p/\partial x & \partial q/\partial x \\ \partial p/\partial y & \partial q/\partial y \end{pmatrix} \equiv \begin{pmatrix} \partial^2 h/\partial x^2 & \partial^2 h/\partial x \partial y \\ \partial^2 h/\partial x \partial y & \partial^2 h/\partial y^2 \end{pmatrix}.$$

Since the image Hessian is a symmetric two-by-two matrix, it is determined by three independent parameters h_{xx}, h_{xy} and h_{yy}.

Define the *Gauss map* to be a function N from a smooth surface S to the unit 2-sphere S^2 mapping each point on S to its respective tangent plane unit normal. That is

$$N: S \to S^2 \quad S^2 = \{(x,y,z) \in R^3 : \|(x,y,z)\| = 1\}.$$

The Gauss map for a smooth surface S is equivalent to the normal map for S using unit normals. Given a differentiable mapping F between two smooth surfaces S_1 and S_2 the *differential*, dF, of F is a mapping of tangent vectors from the tangent plane at each point p on S_1 to the tangent vectors of the tangent plane at F(p) on S_2. The mapping dF is induced by mapping the tangent vectors of parameterized curves going through p on S_1 to the tangent vectors of the image of these parameterized curves under F going through F(p) on S_2. For a formal definition of what it means for a function between two smooth surfaces to be differentiable and a formal definition of the differential of a function see [Do Carmo 1976].

The standard definition of the curvature matrix for the smooth surface S is as the differential dN of the Gauss map N. Therefore the curvature matrix at a point p on S is the two-by-two matrix which when multiplying a vector v in the tangent plane $T_p(S)$ at p gives the rate of change of the surface normal at p along v. For a given parameterization $\phi(x,y)$ a derivation of the curvature matrix is given using the basis vectors ϕ_x and ϕ_y spanning the tangent plane at p. In this case the curvature matrix for S can be represented by

$$dN = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$$

where

$$(2)\quad a_{11} = \frac{fF - eG}{EG - F^2},\quad a_{12} = \frac{gF - fG}{EG - F^2},$$
$$a_{21} = \frac{eF - fE}{EG - F^2},\quad a_{22} = \frac{fF - gE}{EG - F^2}.$$

The terms E,F and G are the coefficients for the first fundamental form and the terms e,f and g are the coefficients for the second fundamental form. Equations (2) are called the *Weingarten equations*. It turns out that the curvature matrix is a self-adjoint linear mapping and that (a_{ij}) can be made symmetric by performing a similarity transformation to an orthogonal basis.

For the image plane parameterization $\phi(x,y) = (x, y, h(x,y))$ the basis vectors spanning the tangent plane at a point are

$$(3)\quad \phi_x = (1,0,h_x)\quad \phi_y = (0,1,h_y)$$

and the first and second fundamental forms are given by

$$(4a)\quad E = 1 + h_x^2,\quad F = h_x h_y,\quad G = 1 + h_y^2,$$

$$(4b)\quad e = \frac{h_{xx}}{\sqrt{1 + h_x^2 + h_y^2}},\quad f = \frac{h_{xy}}{\sqrt{1 + h_x^2 + h_y^2}},\quad g = \frac{h_{yy}}{\sqrt{1 + h_x^2 + h_y^2}}.$$

The Gaussian curvature K is defined to be the determinant of the curvature matrix. Since the curvature matrix is a self-adjoint linear mapping it has two orthogonal eigenvectors which are termed the principal directions of curvature. The respective eigenvalues k_1 and k_2 are termed the principle curvatures with $K = k_1 * k_2$. The mean curvature is given by $H = 1/2(k_1 + k_2)$. The derivation of the Gaussian curvature and the mean curvature using the image plane parameterization above can be found in [Do Carmo 1976]. These are given by

$$K = \frac{h_{xx} h_{yy} - h_{xy}^2}{(1 + h_x^2 + h_y^2)^2},$$

$$H = \frac{(1 + h_x^2) h_{yy} - 2 h_x h_y h_{xy} + (1 + h_y^2) h_{xx}}{(1 + h_x^2 + h_y^2)^{3/2}}.$$

From the expression for K above it is clear that the determinant of the image Hessian has the same sign as the Gaussian curvature and that both are either zero or non-zero simultaneously. Therefore the determinant of the image Hessian gives a good qualitative measure of the Gaussian curvature. While the Hessian matrix is not truly the standard curvature matrix, its components can be substituted into equations (2) and (4) along with h_x and h_y obtained from the determination of local surface orientation. The expressions for K and H are uniquely determined from this information. In section 4 it will be seen that lines of curvature can also be generated from this given information. Note from equations (4b) that the Hessian matrix coincides with the second fundamental form at critical points, that is points where $h_x = h_y = 0$. Also, the determinant of the Hessian matrix is equal to the Gaussian curvature at critical points.

3. DETERMINATION OF THE IMAGE HESSIAN FROM PHOTOMETRIC STEREO

The image Hessian is determined from photometric stereo in analogous fashion to how conventional photometric stereo determines surface orientation. Surface orientation is determined from the intersection of equi-reflectance curves in gradient space. Each equi-reflectance curve arises from the measurement of image irradiance in each successive image. It was shown in [Woodham 1980] that for a Lambertian reflectance map at least three successive images each using a light source at a different incident orientation is required to uniquely determine any surface orientation. Once the surface orientation (p,q,-1) is known, the reflectance gradient (R_p, R_q) is also known. Substituting the image gradient value (I_x,I_y) from a single image into vector equation (1) in section 2 gives rise to the intersection of two planes in a 3-dimensional feature space spanned by h_{xx}, h_{xy} and h_{yy}. The components of the image Hessian are therefore constrained by the equation of a line.

Since the image plane remains stationary for successive images in photometric stereo the components of the image Hessian can be further constrained by using the image gradient (I_x',I_y') from a second image knowing the reflectance gradient (R_p',R_q'). The prime notation for R'(p,q) refers to the same reflectance function R(p,q) but using the different light source incident orientation which is used for the second image. The vector equation

$$\begin{pmatrix} I_x' \\ I_y' \end{pmatrix} = \begin{pmatrix} \partial p/\partial x & \partial q/\partial x \\ \partial p/\partial y & \partial q/\partial y \end{pmatrix} \begin{pmatrix} R_p' \\ R_q' \end{pmatrix}$$

is combined with equation (1) to give four sets of three simultaneous planar equations each with a point solution. Ideally all point solutions should be identical. Since empirical measurements have an inherent error, the point solutions are likely to form a cluster and a statistical method can be used to select the best point value associated with the cluster. If n>2 successive images are used only sets of three simultaneous equations containing all three variables h_{xx}, h_{xy} and h_{yy} produce point solutions. This results in a total of C(2n,3)-2C(n,3) point solutions forming a cluster.

4. LINES OF CURVATURE FROM PHOTOMETRIC STEREO

A connected curve C on a smooth surface S is a *line of curvature* of S if for each point p that C passes through the tangent line is a principal direction of curvature at p on S. Lines of curvature accentuate the shape of a smooth object and provide a good contour representation. It will be shown here how a line of curvature can be traced using the curvature matrix. Since it was shown in sections 2 and 3 how the curvature matrix could be directly recovered from photometric stereo, the lines of curvature can be derived from this method.

To obtain the two orthogonal eigenvectors of the matrix (a_{ij}), the characteristic roots must first be derived. As mentioned in section 2 these are the principal curvatures k_1 and k_2. Letting k represent any one of the principal curvatures and I be the identity matrix, the vector v in the tangent plane of a point p on S is an eigenvector if

$$dN(v) = -kv = -kIv .$$

The minus sign on k results from the definition of the second fundamental form II_p at point p defined by $II_p(v) = -\langle dN_p(v), v\rangle$. Hence the matrix dN+kI cannot be invertible and must have determinant zero. Therefore,

$$det\begin{pmatrix} a_{11}+k & a_{12} \\ a_{21} & a_{22}+k \end{pmatrix} = 0$$

resulting in the characteristic equation

$$k^2 + k(a_{11}+a_{22}) + a_{11}a_{22} - a_{21}a_{12} = 0 .$$

Since the sum of the roots is equal to negative the coefficient of the linear term and $H=1/2(k_1+k_2)$, the characteristic equation can be written

$$k^2 - 2Hk + K = 0$$

with roots

$$k = H \pm \sqrt{H^2 - K} .$$

The eigenvectors $l^{(i)}$, i=1,2 solve the homogeneous system

$$\begin{pmatrix} a_{11}+k_i & a_{12} \\ a_{21} & a_{22}+k_i \end{pmatrix} \begin{pmatrix} l_1^{(i)} \\ l_2^{(i)} \end{pmatrix} = 0 .$$

Hence $l^{(i)}$ can be any scalar multiple of the vector

$$\left(-\frac{a_{12}}{a_{11}+k_i},\ 1\right) .$$

The principal directions of curvature in 3-dimensions are given by

$$(5)\quad l_1^{(i)}\phi_x + l_2^{(i)}\phi_y = (l_1^{(i)}, l_2^{(i)}, l_1^{(i)}h_x + l_2^{(i)}h_y)$$

using equations (3) in section 2. To generate a line of curvature starting at a point p on S select a principal direction of curvature. The projection of this direction onto the image plane is ($l_1^{(i)}$, $l_2^{(i)}$). Proceed for a distance Δs in the image plane along this vector until another sampled image point p' is reached. If the image coordinates for point p are (x_p, y_p), the image coordinates for the new image point p' will be

$$\left(x_p + \frac{l_1^{(i)}}{\sqrt{(l_1^{(i)})^2 + (l_2^{(i)})^2}}\Delta s,\ y_p + \frac{l_2^{(i)}}{\sqrt{(l_1^{(i)})^2 + (l_2^{(i)})^2}}\Delta s\right) .$$

At p' select the principal direction of curvature which is closest to the principal direction of curvature from the previous sampled point and continue the above process from point to point. Since the principal directions of curvature are orthogonal there should be no

problem selecting the proper directions at each point for a reasonably well sampled image unless a line of curvature has a very severe "bend". In this way a map of tangent vectors can be generated for a line of curvature using equation (5). The tangent vectors should be normalized according to

$$\sqrt{(l_1^i)^2 + (l_2^i)^2} = \Delta s$$

for each Δs between sample points in the image. It is then straightforward to construct the actual line of curvature in 3-dimensions up to vertical translation from the image plane.

5. CONCLUSION

It has been demonstrated that the curvature matrix for a smooth surface can be directly derived from a photometric stereo method that utilizes the measurement of image irradiance and the image gradient. The key difference between this method and previous methods of photometric stereo is the use of image gradient values between successive images to completely recover the image Hessian matrix for an arbitrary smooth surface. Previous derivations of the image Hessian matrix required auxiliary assumptions about the smooth surface.

The relationship between the image Hessian matrix and the standard curvature matrix was clarified. The determinant of the image Hessian provides an accurate measure of the sign of the Gaussian curvature. Therefore the photometric stereo method presented here deriving the image Hessian is a very efficient way of determining local convexity, concavity or flatness without having to refer to the normal map. It was shown that the image Hessian could be used together with knowledge of local surface orientation to determine the curvature matrix for an arbitrary smooth surface.

With the curvature matrix derived it was shown how to generate values for the Gaussian and mean curvatures. A method was given for determining lines of curvature which provide a good contour representation for a smooth surface.

REFERENCES

[Do Carmo 1976] Do Carmo, M.P.
Differential Geometry of Curves and Surfaces
Prentice-Hall, 1976

[Woodham 1980] Woodham, R.J.
Photometric Method for Determining Surface Orientation from Multiple Images
Optical Engineering, vol.19 #1 pp.139-144, Jan-Feb 1980

[Woodham 1981] Woodham, R.J.
Analysing Images of Curved Surfaces
Artificial Intelligence, vol. 17 pp.117-140, August 1981

[1]This research was supported in part by ARPA grant #N00039-84-C-0165.

QUALITATIVE INFORMATION IN THE OPTICAL FLOW

Alessandro Verri and Tomaso Poggio

Massachusetts Institute of Technology
Artificial Intelligence Laboratory

ABSTRACT

In this paper we show that the optical flow, a 2-D field that can be associated with the variation of the image brightness pattern, and the 2-D motion field, the projection on the image plane of the 3-D velocity field of a moving scene, are in general quantitatively different, unless very special conditions are satisfied. The optical flow, therefore, is ill-suited for computing structure from motion and for reconstructing the 3-D velocity field, problems that require an accurate estimate of the 2-D the 2-D motion field. As a consequence, we argue that the optical flow should be used to yield information of a more qualitative type, such as motion discontinuities. We also show how the (smoothed) optical flow and 2-D motion field, interpreted as vector fields tangent to flows of planar dynamical systems, have the same qualitative properties from the point of view of the theory of structural stability of dynamical systems.

1. INTRODUCTION

A key task for many vision systems is to extract information from a sequence of images. This information can be useful to solve important problems such as recovering the 3-D velocity field, or segmenting the image into parts corresponding to different moving objects, or reconstructing the 3-D structure of surfaces. The recovery of the 2-D *motion field*, (that we define as the projection on the image plane of the 3-D velocity field) is thought to be an essential step in the solution of these problems. The data available, however, are temporal variations in the brightness pattern. These variations are usually associated with a perceived motion field, called *optical flow* (Gibson, 1950; Fennema and Thompson, 1979; Horn and Schunck, 1981). In order to recover the 2-D motion field, the assumption that the 2-D motion field and the optical flow coincide has often been made. It must be noted though, that this assumption is clearly satisfied only in the case in which variations in the brightness pattern correspond to markings on the visible, 3-D surfaces. In fact several authors have developed algorithms to reconstruct the 2-D motion field from optical flow data defined only at locations of features in the image (Hildreth, 1984a,b; Waxman, 1986). Examples in which this assumption does not hold are known (Horn, 1986), but they have been regarded as pathological cases. As a matter of fact, algorithms that deal with the recovery of the 2-D motion field from dense optical flow data have been proposed, with the more or less explicit assumption that the two fields are the same (Horn and Schunck, 1981; Nagel, 1984; Kanatani, 1985).

In this paper we show that the optical flow and the motion field are in general different, unless very special conditions are satisfied. We explicitly compute the difference between their normal components (the component along the direction of the gradient) under broad assumptions. We show that they are arbitrarily close where the image gradient is sufficiently strong. Hence, feature-based matching algorithms that rely on edges of various types (including texture edges) are more appropriate than point-to-point ones to solve problems that rely on accurate recovery of the 2-D motion field, such as structure from motion. One may then ask, what is the optical flow for? In the second part of the paper we suggest that meaningful information about the 3-D velocity field and the 3-D structure can be obtained from qualitative properties of the 2-D motion field, such as its discontinuities. We then argue that this information can be retrieved directly from the optical flow or its normal components. As an example, we describe a specific approach that exploits results from the theory of stability of dynamical systems. A more detailed analysis of this approach will be presented in a forthcoming paper by V. Torre and coworkers.

The paper is divided in two parts. In the first, we define the problem and we state explicitly the assumptions that we have used. In particular, we consider in detail how image irradiance can be related to scene radiance in the case of a scene consisting of non-lambertian surfaces. We describe, then, a method that

allows us to show that the optical flow and the motion field are almost always different. We compute the difference between the normal components of the two fields assuming, first, the lambertian model of reflectance and then a more realistic one for arbitrary rigid motion of a generic surface. We also calculate how this difference depends on the image gradient and the 3-D velocity of moving objects. In the second part we show how both the optical flow and the motion field can be processed to become vector fields tangent to flows of dynamical systems. The optical flow then, can be considered as a perturbed motion field under the conditions determined in the first part. Results from the theory of stability of dynamical systems suggest that qualitative, stable properties of the motion field hold for the optical flow. We sketch some example of these properties and how they can be used in a description of the 3-D velocity field. We finally discuss briefly some connections with biological systems.

2. PRELIMINARIES

In this chapter we review the definitions of motion field and optical flow, and we state the assumptions that we used throughout the paper. In particular, we consider in detail how image irradiance can be related to scene radiance in the case of a scene consisting of non-lambertian surfaces.

2.1. Definitions

Let us define notations and summarize definitions that will be useful in what follows. Throughout the following we will assume, if it is not otherwise stated, that any expression can be differentiated as many times as nedeed. Let

$$\vec{x}_p = \frac{f}{f + \vec{x}\cdot\vec{n}}(\vec{x} - (\vec{x}\cdot\vec{n})\vec{n}) \qquad (2.1.1)$$

be the equation defining the projection of a generic point on the image plane, where $\vec{x}_p = (x_p, y_p, 0)$ is the position vector of the projected point, $\vec{x} = (x, y, z)$ is the position vector of the point, $\vec{n}$ is the unit vector normal to the image plane (projection plane) and f is the focal length (see Figure 1). Notice that the origin O is on the image plane, the focus of projection F is located at $(0, 0, -f)$, and $f\vec{n} + \vec{x}$ is the vector pointing from F to the point.

The motion field $\vec{v}_p$ can be obtained differentiating

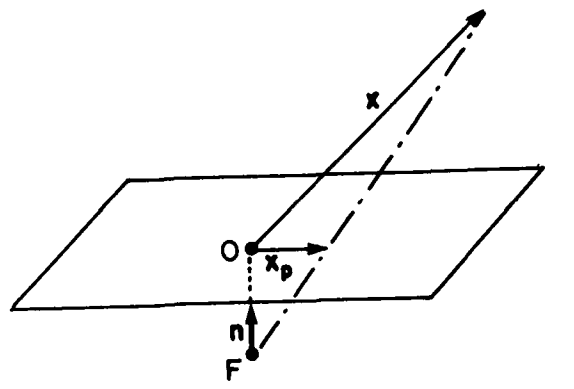


Figure 1. The geometry of perspective projection.

(2.1.1) with respect to the time. If $\vec{v} = d\vec{x}/dt$ we have [1]

$$\vec{v}_p = \frac{f}{f + \vec{x}\cdot\vec{n}}(\vec{v} - (\vec{v}\cdot\vec{n})\vec{n} - \frac{\vec{v}\cdot\vec{n}}{f + \vec{x}\cdot\vec{n}}(\vec{x} - (\vec{x}\cdot\vec{n})\vec{n})) \qquad (2.1.2)$$

Notice that in (2.1.2) $\vec{v}_p$ is given in terms of $\vec{x}$ and $\vec{v}$, position and velocity of the moving points in the scene, which are not known.

Let $E = E(x_p, y_p, t)$ be the image irradiance, that is the intensity of light at the point (x_p, y_p) of the image plane at the time t. If ∇_p is the gradient with respect to the image coordinates, then

$$\frac{dE}{dt} = \frac{\partial E}{\partial t} + \nabla_p E \cdot \vec{v}_p \qquad (2.1.3)$$

Now if

$$\frac{dE}{dt} = 0 \qquad (2.1.4)$$

then, if $\|\nabla_p E\| \neq 0$

$$-\frac{\partial E/\partial t}{\|\nabla_p E\|} = \frac{\nabla_p E \cdot \vec{v}_p}{\|\nabla_p E\|} \qquad (2.1.5)$$

Therefore, if (2.1.4) holds, the projection of the motion field along the direction of the gradient can be given in terms of derivatives of the image irradiance (which can be computed). In what follows, this component will be called $\vec{v}_\perp$, or the *normal component*; thus

$$\vec{v}_\perp = \frac{\vec{v}_p \cdot \nabla_p E}{\|\nabla_p E\|}\frac{\nabla_p E}{\|\nabla_p E\|} \qquad (2.1.6)$$

[1] It can be easily shown that the perspective projection of the 3-D velocity vector is equal to the velocity of the projected point on the image plane, since both the vector are defined in terms of infinitesimal. This is not true for a generic, *finite* vector

Equation (2.1.5) can be interpretated as an instance of the well-known *aperture* problem (Marr and Ullman, 1981; Horn and Schunck, 1981): the information available at each point of a sequence of frames is only the component of the motion field along the direction of the image gradient. To recover a full and unique motion field, some other constraint is needed: Horn and Schunck (1981), for example, showed that there is only one 2-D field whose normal component coincides with (2.1.6) and which is the smoothest of all possible ones. Examples for which (2.1.5) is not true are well known (Horn and Schunck, 1981). Consider, for instance, a rotating sphere with no texture on it (*i.e.* with uniform albedo) under arbitrary, fixed illumination. Since the image irradiance at each image location does not change with time, the left-hand side of (2.1.5) is identically equal to zero, while the right-hand side is different from zero almost everywhere. Notice that keeping the sphere fixed and moving the light source (2.1.5) is again wrong. In this case, however, the left-hand-side is different from zero while $v_\perp$ is zero everywhere. In both cases the perceived motion in the image is different from the motion field. It is worthwhile, then, to introduce a new field, called the *minimal optical flow*, related to the perceived motion in the image, and not necessarily equal to the normal component of the motion field. Notice that the perceived motion in the examples above agrees qualitatively with the left-hand-side of (2.1.5). Indeed, the optical flow in the first case is identically equal to zero, while in the second is different from zero almost everywhere. Therefore let us *define* the normal component $\vec{O}_F$ of the optical flow as:

$$\vec{O}_F = -\frac{\partial E/\partial t \cdot \nabla_p E}{\|\nabla_p E\| \, \|\nabla_p E\|} \qquad (2.1.7)$$

Hence, with respect to this definition, the minimal optical flow and the normal component of the motion field are always directed along the gradient and they coincide if and only if (2.1.4) holds.

Remark: in the literature, it is usually *assumed* that (2.1.4) holds. *When* this is true, the normal components of the motion field and the minimal optical flow are the same and the latter *can* be used as a constraint to recover the 2-D motion field.

2.2. Scene Radiance and Image Irradiance

Let us review briefly some definitions of photometry and make explicit the constraints under which the image irradiance is related to the scene radiance. The image irradiance E is the power per unit area of light at each point (x_p, y_p) of the image plane: thus $E = E(x_p, y_p)$. The scene radiance L is the power per unit area of light that can be thought emitted by each point of a surface S in the scene in a particular direction. This surface can be fictitious, or it may be the actual radiating surface of a light source, or the illuminated surface of a solid. The scene radiance can be thought as a function of the point of the surface and of the direction in space. If (a, b) are intrinsic coordinates of the surface and (α, β) polar coordinates determining a direction in space with respect to the normal to the surface, we can write $L = L(a, b, \alpha, \beta)$. Given the scene radiance it is possible, in principle, to compute the expected image irradiance.

Assuming that the surface is lambertian, *i.e.* $L(a, b, \alpha, \beta) = L(a, b)$, that there are not losses within the system and that the angular aperture (on the image side) is small it can be proved (Born and Wolf, 1959) that

$$E(x_p(a, b), y_p(a, b)) = L(a, b)\Omega \cos^4 \varphi \qquad (2.2.1)$$

where Ω is the solid angle corresponding to the angular aperture and φ is the angle between the principal ray (that is the ray passing through the center of the aperture) and the optical axis. With the further assumption that the aperture is much smaller than the distance of the viewed surface, the lambertian hypothesis can be relaxed to give (Horn and Sjoberg, 1979)

$$E(x_p(a, b), y_p(a, b)) = L(a, b, \alpha^0, \beta^0)\Omega \cos^4 \varphi \qquad (2.2.2)$$

where α^0 and β^0 are the polar coordinates of the direction of the principal ray. It must be pointed out that (2.2.2) holds if L is continuous with respect to α and β. In what follows we will assume that this is the case. Furthermore, we will assume that the optical system has been calibrated. Finally, notice that

$$\nabla_p E \cdot \vec{v}_p = \nabla_S L \cdot \left(\frac{da}{dt}, \frac{db}{dt}\right), \qquad (2.2.3)$$

where ∇_S is the gradient with respect to the surface coordinates, since differentiating (2.2.1) we have

$$\nabla_p E \cdot (dx_p, dy_p) = \nabla_S L \cdot (da, db). \qquad (2.2.4)$$

3. MINIMAL OPTICAL FLOW AND MOTION FIELD

We describe a general method that allows us to show that the minimal optical flow and the normal component of the motion field are almost always different, or

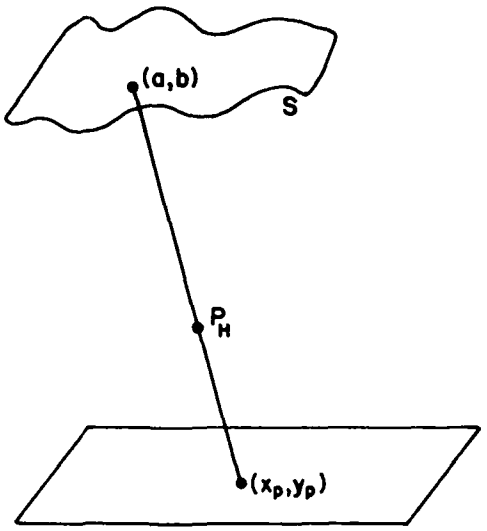


Figure 2. Scene radiance and image irradiance in the pinhole approximation: the image irradiance at the point (x_p, y_p) is given by the scene radiance at the point (a, b) on the surface in the direction of the line connecting the two points and passing through the pinhole P_H.

equivalently that (2.1.4) does not hold. We compute the difference between the normal components of the two fields, assuming first the Lambertian model of reflectance and then a more realistic one for pure translation, pure rotation and general rigid motion of a generic surface. It turns out that the two fields are equal only under very special conditions, which can be explicitly stated. We also show that the difference is smaller where the image gradient is stronger, justifying the use of feature-based algorithms. Of course, this argument does not imply that feature-based algorithms *should* be used: it says, however, that locations of edges (meant here as sharp changes in intensity) contain most of the correct information.

3.1. Computing the Minimal Optical Flow

Consider a rigid surface S moving in space from (2.2.1). The image irradiance E at the time t at the point (x_p, y_p) is equal to the scene radiance L at the point (a, b) on S, *i.e.* $E(x_p, y_p, t) = L(a, b)$. The image irradiance at the time $t + \Delta t$ is given by the scene radiance of the surface at the time $t + \Delta t$. As shown in Figure 3, the point on S that radiates toward (x_p, y_p) at the time $t + \Delta t$ is the point $(a - \Delta a, b - \Delta b)$.[2] The normal $\vec{N}$ to S at the time $t + \Delta t$ at the point $(a - \Delta a, b - \Delta b)$, $\vec{N}_{t+\Delta t}(a - \Delta a, b - \Delta b)$, will be

$$\vec{N}_{t+\Delta t}(a - \Delta a, b - \Delta b) = \vec{N}_t(a - \Delta a, b - \Delta b) + \Delta\vec{N} \tag{3.1.1}$$

where $\Delta\vec{N}$ is the first order variation of $\vec{N}$ due to the motion of S during the time interval Δt. Now in the case of translation

$$\Delta\vec{N} = 0 \tag{3.1.2}$$

while in case of rotation with angular velocity $\boldsymbol{\omega}$

$$\Delta\vec{N} = \boldsymbol{\omega} \times \vec{N}\Delta t \tag{3.1.3}$$

Notice that (3.1.3) can be considered as the expression of $\Delta\vec{N}$ for any kind of motion. Similarly, for each argument A of the scene radiance, we can write

$$A_{t+\Delta t}(a, b) = A_t(a, b) + \Delta A. \tag{3.1.4}$$

To compute ΔA, let us distinguish between arguments of L that are intrinsic function of the surface coordinates (a, b), such as texture and albedo, and those that are in fact function of the space coordinates (x, y, z), (such as the illumination and the point of view) and that are expressed in terms of (a, b) only for convenience. If A is an intrinsic function of the surface coordinates, it follows immediately that

$$\Delta A = 0, \tag{3.1.5}$$

while if A is a function of the space coordinates, from the Taylor expansion we have

$$\Delta A = \nabla A \cdot \vec{v}\Delta t, \tag{3.1.6}$$

where ∇ is the gradient operator with respect to the space coordinates. Let us assume that L can be written as a function of m arguments A^i, $i = 1, ..., m$ and of $\vec{N}$. Then, taking into account (3.1.3) and (3.1.4), (2.2.1) becomes

$$E(x_p, y_p, t + \Delta t) = L\big(A_t^i(a - \Delta a, b - \Delta b) + \Delta A^i, \vec{N}_t(a - \Delta a, b - \Delta b) + \Delta\vec{N}\big) \tag{3.1.7}$$

at time $t + \Delta t$ and

$$E(x_p, y_p, t) = L\big(A_t^i(a, b), N_t(a, b)\big) \tag{3.1.8}$$

at time t. Therefore, using (3.1.6) and (3.1.7),

[2] We assume that the surface corresponds to a moving convex body to avoid self-occlusions due to the motion. In fact, the computation that follows holds for any convex surface patch.

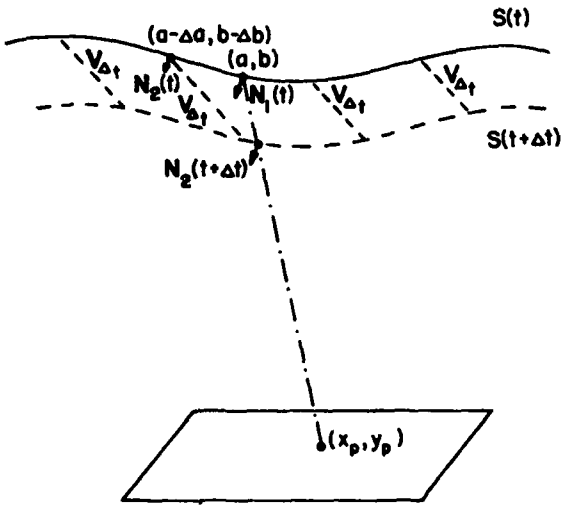


Figure 3 Computing the minimal optical flow: the point (a,b) on S radiates toward (x_p, y_p) at time t. The point $(a-\Delta a, b-\Delta b)$ radiates toward the same point at time $t+\Delta t$. The normal $\vec{N}_1$ is the normal to the S at the point (a,b) and $\vec{N}_2$ at $(a-\Delta a, b-\Delta b)$.

$$\frac{\partial E}{\partial t} =$$

$$\lim_{\Delta t \to 0} \frac{1}{\Delta t}\big(L(A_t^i(a-\Delta a, b-\Delta b) + \Delta A^i,$$

$$\vec{N}_t(a-\Delta a, b-\Delta b) + \Delta \vec{N}) - L(A_t^i(a,b), N_t(a,b))\big), \tag{3.1.9}$$

where the ΔA^i are computed using (3.1.5) or (3.1.6) according to the kind of argument. From (3.1.9), the minimal optical flow can be derived easily. To simplify notation, let us suppress the subscript t from Equation (3.1.9).

From (3.1.9) we easily get

$$\frac{\partial E}{\partial t} = -\nabla_p L \cdot \left(\frac{da}{dt}, \frac{db}{dt}\right) + \sum_{i=1}^{p} \frac{\partial L}{\partial A_i} \nabla A^i \cdot V + \frac{\partial L}{\partial N} \omega \times N$$

if p of the $A^i (i = 1, \ldots, m)$ require the use of (3.1.6) to compute ∇A and $\frac{\partial L}{\partial N} = \left(\frac{\partial L}{\partial N_x}, \frac{\partial L}{\partial N_y}, \frac{\partial L}{\partial N_z}\right)$ for $N = (N_x, N_y, N_z)$.

Therefore, using (2.1.6), (2.1.7), and (2.2.4), we can write

$$v_\perp - 0_F = \sum_{i=1}^{p} \frac{\partial L}{\partial A_i} \nabla A^i \cdot v + \frac{\partial L}{\partial N} \cdot \omega \times N \tag{3.1.10}$$

Thus, the normal components of the two fields are different if the surface undergoes a motion with a rotational component, or the reflectance function contains arguments depending on space coordinates.

Let us consider now some interesting examples in detail.

3.2. Translation of a Lambertian Surface

Consider a lambertian surface S. The scene radiance due to S will be

$$L = \rho \vec{I} \cdot \vec{N} \tag{3.2.1}$$

where ρ is the albedo of S, $\vec{I}$ the unit vector in the direction of the illumination and $\vec{N}$ is the unit normal to the surface. Let us compute the difference (3.1.10) between the normal components of the optical flow and of the motion field corresponding to a translation of S in space with velocity $\vec{v}$ under uniform fixed illumination. Substituting (3.2.1) in (3.1.10) and changing the sign, we have

$$v_\perp = O_F, \tag{3.2.2}$$

since $\omega = 0$ and none of the outputs of L in (3.2.1) depends on space constraints (I is constant). Therefore, the minimal optical flow of a translating lambertian surface uniformly illuminated is exactly equal to the motion field.

Remark: in the case of non-uniform illumination the right hand side of (3.2.2) contains an extra term due to $\Delta \vec{I}$. Using (3.1.6) to compute the components of $\Delta \vec{I}$, (3.1.10) yields

$$v_\perp - O_F = \frac{1}{\|\nabla_p E\|} \rho \left(\frac{\partial \vec{I}}{\partial x}\frac{dx}{dt} + \frac{\partial \vec{I}}{\partial y}\frac{dy}{dt} + \frac{\partial \vec{I}}{\partial z}\frac{dz}{dt}\right) \cdot \vec{N},$$

which can be rewritten

$$v_\perp - O_F = \frac{1}{\|\nabla_p E\|} \rho \frac{d\vec{I}}{dt} \cdot \vec{N}, \tag{3.2.3}$$

since $\partial \vec{I}/\partial t = 0$ (the illumination is supposed to be fixed).

3.3. Rotation of a Lambertian Surface

Let S be a lambertian surface rotating in space with angular velocity ω. Let $\vec{I}$ be again uniform. Applying the same argument of the previous section but taking into account the constraint (3.1.3) for ∇N, we get

$$v_\perp - O_F = \frac{\rho \vec{N} \cdot \vec{I} \times \omega}{\|\nabla_p E\|}. \qquad (3.3.1)$$

In the case of rotation, therefore, even under uniform illumination, the minimal optical flow and the normal component of the motion field are different. They are equal for any surface only if ω and $\vec{I}$ are parallel. This corresponds to the case of a surface rotating around an axis parallel to the direction of uniform illumination. In the case of non-uniform illumination, an extra term like the one in (3.2.3) must be added to (3.3.1).

3.4. Translation of a Specular Surface

Let us consider now a model of reflectance more realistic than the lambertian one. Following Phong (1975; see also Horn and Sjoberg, 1979) we define the scene radiance as a linear combination of a lambertian and a specular term, *i.e.*

$$L = L_{\text{lamb}} + L_{\text{spec}}. \qquad (3.4.1)$$

The lambertian term is equal to the one used before, while the specular term is

$$E = \frac{s\vec{D} \cdot \vec{R}}{D}, \qquad (3.4.2)$$

where s is the fraction of light reflected by the surface, $\vec{D} = f\vec{n} + \vec{x}$ is the vector pointing from the focus to the radiating point and

$$\vec{R} = \vec{I} - 2(\vec{I} \cdot \vec{N})\vec{N} \qquad (3.4.3)$$

is the unit vector in the direction of the perfect specular reflection. Let us assume that s is not a function of the direction of the incident light and that it is constant on the surface. The specular term is thus proportional to the cosine of the angle between the direction of specular reflection and the line of sight.

Since we are computing derivatives and L is a linear combination of L_{lamb} and L_{spec} we can compute separately the contributions to the minimal optical flow due to the lambertian and the specular term, adding the results afterward. Therefore, we only need to compute now the specular one. Let us consider, first, the case of pure translation of a surface S radiating accordingly to (3.4.2) and let us call S a *specular* surface. If S is translating with velocity $\vec{v}$ and $\vec{I}$ is uniform, substituting (3.4.2) into (3.1.10) and taking into account the constraint (3.1.2), we have

$$v_1 - 0_F \frac{s}{D^3} \frac{(D^2\vec{v} \cdot \vec{R} - (\vec{D} \cdot \vec{v})(\vec{D} \cdot \vec{R}))}{\|\nabla_p E\|}, \qquad (3.4.4)$$

since from (3.1.6)

$$\lim_{\Delta t \to 0} \frac{\Delta \vec{D}}{\Delta t} = \frac{\partial \vec{D}}{\partial x}\frac{dx}{dt} + \frac{\partial \vec{D}}{\partial y}\frac{dy}{dt} + \frac{\partial \vec{D}}{\partial z}\frac{dz}{dt} = \frac{d\vec{D}}{dt} = \frac{d\vec{x}}{dt} = \vec{v}. \qquad (3.4.5)$$

Using again the two fields we get a well known vector identity:

$$v_\perp - O_F = \frac{s}{D^3} \frac{(\vec{v} \times \vec{D}) \cdot (\vec{R} \times \vec{D})}{\|\nabla_p E\|}. \qquad (3.4.6)$$

Thus, in the case of translation of a specular surface, the minimal optical flow and the normal component of the motion field are always different.

3.5. Rotation of a Specular Surface

Consider now the same *specular* surface S rotating in space with angular velocity ω. Then, substituting (3.4.2) into (3.1.10) and taking into account the constraint (3.1.3), we have, after some algebra,

$$v_\perp - O_F = \frac{s}{D^3\|\nabla_p E\|}\Big(D^2(\vec{I} \times \omega) \cdot (\vec{D} - 2(\vec{D} \cdot \vec{N})\vec{N}) - f(\vec{n} \times \omega) \cdot (\vec{D} \times (\vec{D} \times \vec{R}))\Big). \qquad (3.5.1)$$

The minimal optical flow, therefore, is equal to the motion field for any specular surface only when $\vec{I}$, ω and $\vec{n}$ are parallel.

3.6. General Case

Let us consider, now, the general case. We will assume (3.4.1) as scene radiance of a surface S undergoing a given rigid motion (composition of a rotation and a translation) in space. Adding together (3.3.1), (3.4.6) and (3.5.2), we obtain the difference between the motion field and the minimal optical flow for a surface in the general case under uniform illumination, *i.e.*:

$$v_\perp - O_F = \frac{\rho \vec{N} \cdot \vec{I} \times \omega}{\|\nabla_p E\|} + \frac{s}{D^3} \frac{(\vec{v} \times \vec{D}) \cdot (\vec{R} \times \vec{D})}{\|\nabla_p E\|} + \frac{s}{D^3\|\nabla_p E\|}\Big(D^2(\vec{I} \times \omega) \cdot (\vec{D} - 2(\vec{D} \cdot \vec{N})\vec{N}) - f(\vec{n} \times \omega) \cdot (\vec{D} \times (\vec{D} \times \vec{R}))\Big) \qquad (3.6.1)$$

The right-hand side of (3.6.1) is generally different from zero. In fact, there are no general conditions under which it is identically equal to zero. Notice, however, that if ω and v are bounded

$$\lim_{|\nabla_p E| \to \infty} |v_\perp - O_F| = 0, \qquad (3.6.2)$$

Equation 3.6.2 shows that the points in the image where the gradient is stronger are the points where the minimal optical flow is closer to the motion field. These points are characterized by sharp changes in intensity - edges - that usually correspond to important

physical events on surfaces, such as boundaries and orientation discontinuities. Thus, to solve problems such as structure from motion, or the recovery of the 3-D velocity field, which require an accurate, quantitative estimate of the 2-D motion field, edge-based algorithms seem more suitable than algorithms based on spatial and temporal derivatives of the image brightness. As a consequence, in order to obtain a precise reconstruction of the 2-D motion field, algorithms based on the solution of the correspondence problem among suitably defined features (as in stereo) may be used. This argument agrees with the fact that, as intuitively expected, the minimal optical flow and the motion field at image *features* corresponding to precise locations on the 3-D surfaces coincide. It must be pointed out that in this analysis we have not considered shadows and self-shadow effects. They usually give rise to edges in the image that do not correspond to features in the scene. Furthermore, the Phong model of reflectance does not include sharp intensity changes due to specularities.

4. QUALITATIVE PROPERTIES: NOT JUST DISCONTINUITIES

Traditionally, the optical flow has been considered as the first step for recovering 3-D structure and 3-D motion. In this chapter we suggest a different use of the minimal optical flow. We argue that qualitative properties of the 2-D motion field give useful information about the 3-D velocity and the 3-D structure of surfaces and that these qualitative properties can be usefully inferred from the obtainable minimal optical flow. Discontinuities in the optical flow are the typical example of the stable qualitative properties the optical flow can deliver. Motion discontinuities can be computed easily in many cases (Reichardt et al., 1983; see also Little et al., these Proceedings). They are also important for the integration of visual information (Poggio, 1985b). They are stable in the sense that they are likely to be invariant to quantitative changes in the way the optical flow is computed.

In this chapter we discuss another example of stable, qualitative properties of the optical flow. We introduce the qualitative properties associated with 2-D dynamical systems and show how to process minimal optical flow and motion field for making them equivalent to flows of dynamical systems on the plane. We then suggest, from properties of structural stability of dynamical systems, that the minimal optical flow may be equivalent to the motion field in terms of qualitative properties.

4.1. Smoothing the Optical Flow and the Motion Field

In order to establish a connection with the theory of stability of dynamical systems, we must insure that the optical flow and the motion field have an appropriate degree of smoothness. This is not always the case, because of discontinuities arising at object boundaries or because of noise. We suggest to use a filtering step to smooth the field. It is worthwhile noticing that a filtering step on the normal component of a dense motion field is a (regularization) method to recover the whole 2-D motion field. [3]

4.2. Qualitative Descriptions of Dynamical Systems

For a rigorous and thorough review on dynamical systems see Hirsch and Smale (1974). Here, for the sake of completeness, we summarize the main definitions and results.

A *dynamical system* is a C^1 map ϕ: $R \times A \to A$, where A is an open set of an Euclidean space and writing $\phi(t,x) = \phi_t(x)$, the map ϕ_t: $A \to A$ satisfies:

(a) ϕ_0: $A \to A$ is the identity;

(b) the composition $\phi_t(\phi_s(x)) = \phi_{t+s}$ for each $t, s \in R$.

A dynamical system ϕ_t on A gives rise to a differential equation on A, that is a vector field y: $A \to E$ defined as follows:

$$y(x) = \frac{d}{dt}\phi_t(x)\Big|_{t=0} . \qquad (4.3.1)$$

Thus, for every x, $y(x)$ is the tangent vector to the curve $t \to \phi_t(x)$ at $t = 0$. In the following, we will restrict our attention to planar systems, (*i.e.* in what follows, A will be an open set in R^2). Solutions like x^0 are called *equilibrium points* or *equilibria*. The restriction to planar systems reduces the classification to four fundamental cases:

I : The *saddle*: the equilibrium is unstable (an equilibrium is *stable* if any nearby solutions to it stays nearby for all the future time. It is *unstable* otherwise).

[3] This "smoothed" 2-D motion field may not be the same recovered using standard algorithms, but its qualitative properties are likely to be preserved. The analogy we are about to present, indeed, will support this argument (and the equivalence between qualitative properties of the 2-D motion field and the optical flow as well).

II: The *sink* which is a stable equilibrium. The main property of a sink is that

$$\lim_{t\to\infty} x(t) = 0$$

III: The *source*. The main property of a source is that

$$\lim_{t\to\infty} |x(t)| = \infty$$

and

$$\lim_{t\to-\infty} |x(t)| = 0$$

. A source can be considered as the dual case of a sink: the phase portrait of a source and of the correponding sink are the same except that for the direction of the motion which must be reversed.

IV: The *center*. All the solutions are *periodic* with the same period. A center is a stable equilibrium. A center, however, is not a structurally stable property.

The crucial point is that this classification is *exhaustive*. Every solution to Equation (4.3.1) (in the linear case) looks like a saddle, a sink, a source, or a center. The same classification holds for the non-linear case. However non-linear systems are interesting in themselves, since they can show a different qualitative behavior. A non-linear system can have in addition *limit cycles*. Intuitively, a limit cycle is a closed orbit towards which other solutions' curves spiral with the same asymptotic period. Defining a *ω-limit set*, $L_\omega(x)$, as $L_\omega(x) = \{a \in A$ such that $\exists t_n \to \infty$ with $x(t_n) \to a\}$ and similarly an *α-limit set* $L_\alpha(x)$ as $L_\alpha(x) = \{b \in A$ such that $\exists t_n \to -\infty$ with $x(t_n) \to b\}$, a limit cycle is a closed orbit γ such that $\gamma \subset L_\omega(x)$ or $\gamma \subset L_\alpha(x)$ for some $x \neq \gamma$. Under somewhat more restrictive conditions, a limit cycle can be a *periodic attractor* (for a rigorous definition of it, see Hirsch and Smale 1974). Intuitively, a periodic attractor is a limit cycle such that nearby trajectories not only have the same asymptotic period but also are in phase.

Saddles, sinks, sources and periodic attractors are very important for a qualitative description of planar systems. It can be shown that such properties are *structurally stable*, that is they persist after a perturbation of the right-hand side of (4.3.2). As far as planar systems are concerned they also *fully* characterized limit sets. By means of the Poincaré-Bendixon theorem it can be shown that compact limit sets other than limit cycles are saddles, or sinks, or sources or trajectories joining them.

4.3. Equilibria and their Interpretations

In the definition of dynamical system the left-hand side of Equation (4.3.1) can be interpretated as a vector field tangent to the family of curves in the plane, solutions to (4.3.1) itself. It is straightforward to see that both the smoothed optical flow and motion field (i.e. after the filtering operation) can be considered as istances of such a vector field [4]. Indeed, it is sufficient to insure that both the fields are *continuous with continuous first derivatives*. The classification of the solutions can now be interpreted in terms of characteristic points of the 2-D motion field. A source, for example, corresponds to a focus of expansion of the field. The structural stability of the source, in turn, says that a focus of expansion persists even if the field is perturbed. From this perspective a focus of expansion is expected to be detectable in a 2-D motion field reconstructed with different algorithms and in the optical flow as well, when they can be considered as perturbed examples of the "true" 2-D motion field.

4.4. Discussion

If our point of view is correct, the only critical property of the optical flow is that it have the same qualitative properties of the 2-D velocity field. Quantitative equivalence, which is impossible in general, is in any case irrelevant for this use of the optical flow. As a consequence, *many different "optical flows" may be defined.* Equation (2.1.6) does not have any priviliged role: other definitions could be preferred on the basis of criteria such as computability (from image data) or ease of implementation (for given hardware constraints).

This point of view has clear implications for biological visual systems: movement detecting cells (say, in the retina) do not have to compute the specific minimal optical flow defined by equation (2.1.6): other, possibly simpler, estimates of the velocity field that preserve its qualitative properties are equally good candidates (such as correlation-like algorithms). This argument may explain why the models proposed to explain motion dependent behaviour in insects (Hassenstein and Reichardt, 1956), motion perception in humans (Van Santen and

[4] We stress the fact that the analogy with the dynamical system is between *phase portraits* of dynamical systems and motion flows. The parameter t in the definition of dynamical system is *not* the physical time. We considered motion flows, such as the 2-D motion field or the optical flow at a fixed time, comparing them with the vector field tangent to the phase portrait of some system: we are not interested in the physical meaning of the underlying dynamical system.

Sperling, 1984) and physiology of cells (Barlow and Levick, 1965; Torre and Poggio, 1978) are all implementing computations quite different from the minimal optical flow as it is usually defined (see equation 2.1.6). In addition all these models do not typically measure velocity – not even in the case of uniform translation in a frontoparallel plane. Even for simple motions of the latter type the output of models such as the correlation models depends on both the velocity and the spatial structure of the moving pattern. One is tempted to consider this as a weakness of these models compared to the definition of minimal optical flow, Equation 2.1.7. Our results, however, show that this is not the case: first, the minimal optical flow is correct only in a very special situation; second, all these models may have the same qualitative properties of the motion field, which, from our point of view, is the only critical requirement for a "good" measurement of motion. The next question is of course whether these biological models are in fact "close" enough to the motion field to share the same qualitative properties. We do not know the answer yet. We conjecture, however, that they are indeed usually similar enough to preserve the main qualitative properties of the motion field. The conjecture is based on results (Poggio and Reichardt, 1973 and Poggio, 1985b) showing that most of the biological models proposed so far can be considered as special instances or approximations of a general class of nonlinear models (characterized as Volterra systems of the second order); and that the minimal optical flow, as defined in equation , is also approximately a Volterra functional of the second order (Poggio, 1985a).

It is important to stress that the approach outlined in the second part of this paper for classifying the qualitative properties of the optical flow is only one of the possible methods. While we plan to develop further that particular approach, others should be explored as well: in particular flows that do not correspond to dynamical systems on the plane may be better suited for capturing important and stable properties of the velocity field such as motion discontinuities (think of hydrodynamics). In this case, the classification of qualitative properties should take place without a preliminary smoothing operation. This approach would have the advantage of preserving the discontinuities of the field that, as we argued, represent very useful information.

In addition to the *classification* of *stable* qualitative properties of the velocity field, much work needs to be done at the level of their *interpretation* in terms of 3-D structure and 3-D velocity. Some of the qualitative properties of the (smoothed) velocity field have an easy interpretation in those terms: an obvious example is again a focus of expansion that is related to "crashing" motion. It is likely that many, more subtle relations exist between the qualitative properties of the flow and the underlying 3-D motion and structure. For example, preliminary results by Torre et al. (personal communication) suggest that the number of *focuses* in the (smoothed) field may be characteristic for the rigidity of motion in the visible scene.

Finally, we should mention an obvious extension of the approach described in the second part of the paper. We have only considered so far the velocity field "frozen" at a given instant of time. The succession of image frames provides in fact a time-dependent field: the evolution in time of the qualitative properties we have described – how they are created, disappear and transform – should be characterized in qualitative terms, for instance using the language of catastrophe and bifurcation theory. The use of time-dependent fields should be practically much more robust, because of the redundant information available in a sequence of *very closely* spaced frames (in time). Our analysis should be extended to qualitative properties that are structurally stable not only at a given time but also in the time dependent field.

ACKNOWLEDGEMENTS

We would like to thank Vincent Torre, Davi Geiger, B. Caprile, Berthold K.P. Horn, and especially Bror Saxberg for useful discussions.

REFERENCES

Barlow, H.B. and R.W. Levick, (1965) The mechanism of directional selectivity in the rabbit's retina, J. Physiol. **173:** 477–504.

Born, M. and Wolf, E. 1959 *Principles of Optics.* New York: Pergamon Press.

Gibson, J. J. 1950. *The Perception of the Visual World.* Boston: Houghton Mifflin.

Fennema, C. L., Thompson, W. B. 1979. Velocity determination in scenes containing several moving objects. *Comput. Graph. Image Proc.* 9:301–315.

Hassenstein, B. and Reichardt, W. (1956) Systemtheoretische Analyse der Zeit-, Reihenfolgen- und Vorzeichenauswertung bei der Bewgungsperzeption der Russelkafers, Chlorophanus. Z. Naturforsch. IIb: 513–524.

Hildreth, E. C. 1984a. *The Measurement of Visual Motion.* Cambridge: MIT Press.

Hildreth, E. C. 1984b. The Computation of the velocity field. *Proc. R. Soc. London B* 221:189–220.

Hirsch, M.W. and Smale, S. 1974 *Differential Equations, Dynamical Systems, and Linear Algebra.* New York: Academic Press.

Horn, B.K.P. *Robot Vision*, Cambridge and New York: Massachusetts Institute of Technology Press and McGraw-Hill, 1986.

Horn, B. K. P., Schunck, B. G. 1981. Determining optical flow. *Artif. Intell.* 17:185–203.

Horn, B.K.P. and Sjoberg, R.W. 1979. Calculating the reflectance map. *Applied Optics* 18: 1770-1779.

Kanatani, K. 1985. Structure from motion without correspondence: general principle. *Proc. Image Understanding Workshop*, Miami, FL, pp. 107–116.

Marr, D., Ullman, S. 1981. Directional selectivity and its use in early visual processing. *Proc. R. Soc. London Ser. B* 211:151–180.

Nagel, H.–H. 1984. Recent advances in image sequence analysis. *Proc. Premier Colloque Image — Traitement, Synthese, Technologie et Applications*, Biarritz, France, May, pp. 545–558.

Phong, B.T. 1975 Illumination for computed generated pictures. *Communications of the ACM* 18:311-317.

Poggio, T. 1985a Cold Spring harbor lecture, unpublished.

Poggio, T. 1985b "Integrating Vision Modules with Coupled MRF's," Massachusetts Institute of Technology Artificial Intelligence Laboratory Working Paper 285.

Poggio, T., Reichardt, W. 1973. Considerations on models of movement detection. *Kybernetik 13*, 223-227.

Reichardt, W., Poggio, T., Hausen, K. 1983. Figure–ground discrimination by relative movement in the visual system of the fly. Part II: Towards the neural circuitry. *Biol. Cyber.* 46:1–30.

Schunck, B. G., Horn, B. K. P. 1981. Constraints on optical flow computation. *Proc. IEEE Conf. Patt. Recog. Image Proc.*, August, pp. 205–210.

Torre, V. and T. Poggio, (1978) A Synaptic mechanism possibly underlying directional selectivity to motion. Proc. R. Soc. Lond. B **202:** 409–416.

Van Santen, J. P. H., Sperling, G. 1984. A temporal covariance model of motion perception. *J. Opt. Soc. Am. A*, 1:451–473.

AUTHORS

Tomaso Poggio: Artificial Intelligence Laboratory, NE43-787; Massachusetts Institute of Technology; 545 Technology Square; Cambridge, MA 02139 Alessandro Verri: Universita di Genova; Dipartimento di Fisica; Via Dodecanso 33; 16146 Genova, Italy

Algorithm Synthesis for IU Applications

Michael R. Lowry

AI Lab, Stanford University, Stanford, California 94305

Abstract

Computer vision algorithms incorporate substantial domain knowledge and programming knowledge. This paper describes the capabilities needed for automatic programming in the domain of Image Understanding (IU) algorithms. A general model of algorithm derivation is given based upon an analysis of papers in the December 1985 IU workshop proceedings. Parts of this general model are being implemented in the STRATA automatic programming system. The derivation of a non-trivial geometric optimization algorithm, the Karmarker linear optimization algorithm, is given to illustrate the methods used by STRATA.

1 Introduction

Computer vision presents unique opportunities and challenges to automatic programming research. Computer vision algorithms incorporate substantial domain knowledge from areas such as geometry, photometry, and probability. Computer vision is computationally intensive, yet even research prototypes must have reasonable running times in order to adequately test ideas against real images. Thus computer vision research draws upon sophisticated knowledge in numerical analysis, analysis of algorithms, and computational geometry. Finally computer vision is a rapidly developing field with new concepts, theories, methods, and algorithms reported every year.

This paper describes research in progress for partially automating computer vision algorithm development. In the near term, the methods which have been developed could be used to implement an intelligent programming assistant for computer vision programming. The rest of this introduction briefly describes some previous automatic programming research in knowledge intensive domains. It also overviews the methods of parameterized theory instantiation and invariant logic. The second section presents a general model of Image Understanding algorithm derivation with a brief analysis of selected papers from the last IU workshop. The third section is a conceptual overview of the Karmarker linear programming algorithm. This algorithm resembles many of the non-linear optimization algorithms that arise in computer vision. The Karmarker algorithm is also a good illustration of the automatic programming methods I have developed. The last section is a mechanizeable derivation of a Karmarker algorithm variant based upon parameterized theory instantiation and invariant logic.

Most research in automatic programming has been in knowledge-poor domains, where the primary complexity is in algorithm derivation. An exception is the work of Barstow et al [Bar86] on synthesizing programs for the domain of oil exploration at Schlumberger. Barstow finds that the complexity of oil exploration software arises from the wealth of the oil exploration problem domain, not the algorithms employed. In contrast, computer vision software is complex due to both the wealth of the problem domain and the sophistication and variety of the algorithms which are employed. The major challenge of computer vision programming is the interaction of sophisticated domain knowledge and sophisticated algorithm knowledge. The same conclusion has been reached by Kant [KN83] who has examined the domain of computational geometry.

The idea of formalizing high level algorithm schemas, such as iterative convergence and hill climbing, is well known although their use in automatic programming is fairly recent [Smi85] [Ric86]. In my research, the applicability conditions for an algorithm schema are represented declaratively as a *parameterized theory* [GB85]. Actually, the relationship between the abstract input/output conditions, the problem structure used by an algorithm, and the algorithm schema itself is fairly complex and has usually been given a schema specific procedural encoding. A general declarative representation is necessary for cleanly specifying the incorporation of many different schemas into one particular algorithm, and also for facilitating the use of different search techniques in instantiating an algorithm schema. The details of the declarative representation for algorithm schemas can be found in [Low87], this paper illustrates their use in algorithm synthesis through pa-

rameterized theory instantiation. The mathematical theory of parameterized theories has been developed by theoreticians applying mathematical logic to abstract data types [GTW78].

Another area of intensive research in automatic programming has been the automatic selection of efficient data types for set theoretic constructions. This is the concern of the SETL project [SSS86]. Yet most scientific programming that is related to real world physics relies upon the efficient implementation of geometric constructions and operations. This is certainly true of IU (image understanding) and robotics.

My research on automatic programming develops a framework for choosing efficient data types for geometric constructions. This framework is based upon the modern approach to Geometry first proposed by Felix Klein:

> A geometry is the study of the properties of a set S which remain invariant when the elements of a set S are subjected to the transformations of some transformation group.

The role of invariants and transformation groups is central to modern geometry and physics, in fact Einstein at one time called his theory Invariant Theory.

I apply this idea to automatic programming by finding the invariants of a problem for which an algorithm is to be synthesized. The invariants of a problem indicate in what abstract theory a problem should be formulated, and what transformations can be applied in choosing an efficient formulation of the problem. The idea of reformulating a problem using a transformation which leaves the io-behavior invariant is the source of power for the Karmarker linear optimization algorithm, whose synthesis will be described in this paper.

2 Model of Algorithm Derivation for IU applications

The following model of algorithm derivation for image understanding is based upon my own experience in computer vision [LM81] [Fea82] [LM82] [Low82] [WBKL83] and my association with Stanford's vision group. The model will be illustrated with a sampling of those papers in the 1985 IU proceedings whose title indicated an emphasis on algorithms[1] : [PRH85], [Lee85], [CK85]. Parts of this model are being mechanized in my research on automatic programming, while other parts will be studied in future research. Total success in mechanizing this model would automate IU research! Most likely, this research will lead to the development of an intelligent programming assistant for rapid prototyping of IU algorithms and for verifying the correctness of IU algorithms.

The general model of IU Algorithm derivation:

1. Formulate an ill-structured problem as a formal mathematical problem. Examples of ill-structured problems are recovering depth from shading, deriving motion parameters from successive camera frames, or interpolating depth information from sparse data. The formal problem is often formulated as finding a solution to a set of equations, or an optimal solution to a set of equations.

2. Instantiate several known *algorithm schemas* to derive a skeleton algorithm which solves the formal problem. For example, hill climbing, iterative convergence, and coarse-to-fine are all algorithm schemas that were used in the papers mentioned above.

3. The skeleton algorithm is instantiated down to the level of language independent subroutines, such as gaussian elimination or linear least squares. These subroutines would be reasonably expected to be in a scientific library package.

The following papers from last year's IU proceedings illustrate this model of algorithm derivation:

Pavlin [PRH85] analyzed an algorithm for detecting translational motion. The ill-structured problem is to determine vehicle motion using vision. This problem is formalized as an optimization problem, namely minimizing the global error for the parameters of translational motion given correlation of feature in successive camera frames. The top level of the algorithm uses a coarse to fine search to determine the parameters of translational motion. Within each level of coarseness, a hill climbing algorithm is used to find the parameter values with minimal error. The knowledge needed for this algorithm includes probability, geometry, and the search schemas of coarse-to-fine and hill climbing.

Lee [Lee85] presents an iterative algorithm for shape from shading. The ill-structured problem is to determine shape from shading and occluding boundaries. This problem is formalized as an optimization problem over a set of non-linear equations. The paper shows existence and uniqueness of a minimal solution of the non-linear equations, and proves that a new iterative algorithm based on spline-smoothing converges.

Choi [CK85] presents a parallel algorithm for solving the depth interpolation problem. They state in their abstract: "Many low level computer vision problems, including depth interpolation, can be cast as solving a

[1] Thanks to Professor Binford for this suggestion.

symmetric positive definite matrix." First, the depth interpolation problem is formalized as solving a set of linear equations. Because of the special form of the matrix, various iterative methods with fast convergence rates are applicable. The major part of the paper describes the parallelization of an adaptive chebyshev acceleration method, using a a parallel tree architecture.

The first step of formalizing an ill-structured problem involves many creative aspects based on experience and judgement. *Simplifying assumptions* are invoked which hopefully do not exclude the real world problems which motivated the research. This is why testing against real data is the ultimate verification of an IU algorithm. Methods for deriving an algorithm given a formal mathematical specification are described in subsequent sections.

3 Linear Optimization

Linear optimization is finding the optimum value of a linear cost function given a set of linear constraints. As shown in the previous section, mathematical optimization is at the core of many IU algorithms. Sometimes the cost function or the set of constraints is non-linear, but the derivation of these algorithms have much in common with the linear case. The derivation of a variant of the Karmarker algorithm is used to illustrate my methods of automatic programming and problem reformulation. This section presents an overview of linear programming algorithms and the reformulation methods used to synthesize them from specification. The next section presents the details of the algorithm derivation.

A linear cost function over R^N is expressed as a function of the following form:

$a_1x_1 + a_2x_2 + ...a_nx_n$

In linear optimization, the objective is to minimize (or equivalently maximize) this function over a domain which is defined by a set of linear equalities and inequalities:

$a_{11}x_1 + a_{12}x_2 + ...a_{1n}x_n = b_1$

$a_{m1}x_1 + a_{m2}x_2 + ...a_{mn}x_n < b_m$

From the more abstract viewpoint of Euclidean geometry, the linear constraints describe a convex polyhedra (possibly unbounded or null), and the cost function describes a direction. The desired output is the point on the polyhedra which is furthest along the direction vector.

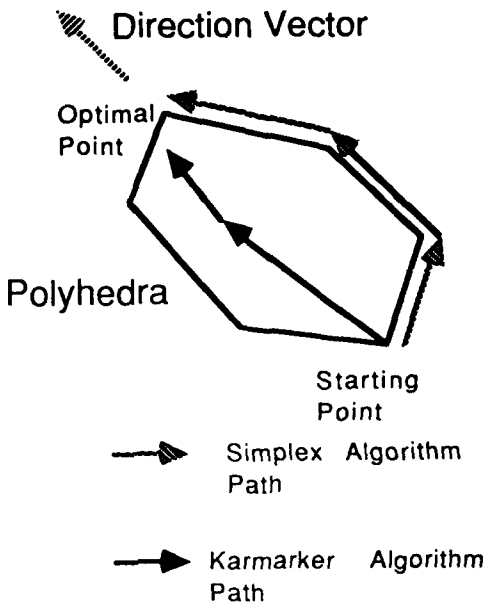
The insight of the simplex algorithm is that the output will be a vertex of this polyhedra (or at least include a vertex, in degenerate cases where an entire edge or plane is perpendicular to the direction vector). The skeleton of the simplex algorithm is hill climbing between adjacent vertices until a local optimum is reached. Because of convexity, a local optimum is guaranteed to be a global optimum.

The insight of the Karmarker algorithm is that the linear optimization problem is abstractly a problem of oriented projective geometry. This means that only incidence (e.g. point p is on line l) and sidedness (e.g. points p and q are on opposite sides of line l) are essential properties for solving linear optimization problems. Distance metrics, angle metrics, even parallelism are irrelevant. In other words, linear optimization is invariant under projective transformations which preserve orientation. The modified Karmarker algorithm is somewhat less general, using the invariance of linear optimization under affine transformations. The difference is that the modified version preserves parallelism, consequently it is not a provably polynomial time algorithm. However, both the algorithm and the derivation are easier to explain, while preserving the flavor of the Karmarker algorithm. On practical problems, the Karmarker algorithm and the modified version have similar performance.

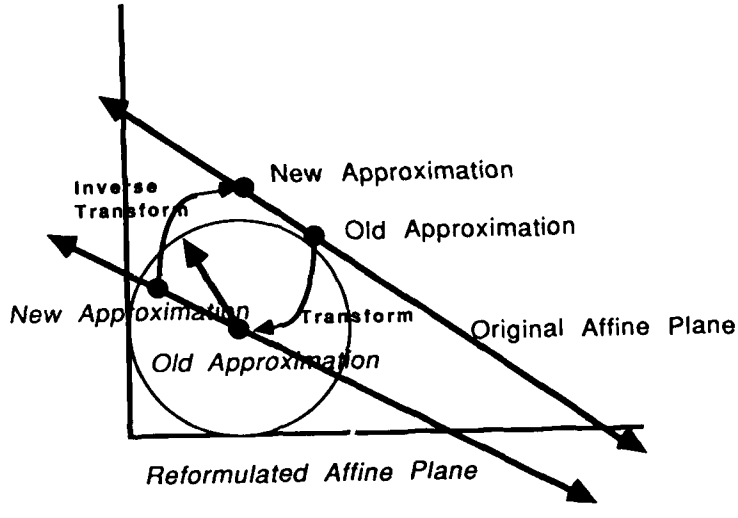
The skeleton algorithm for the Karmarker algorithm and related algorithms that travel through the center of the polyhedra, as opposed to the boundary of the polyhedra, is an iterative approximation algorithm. Each iteration begins with the polyhedra, the direction vector, and an approximate interior optimal point and derives a better interior approximation to the optimal point. Convergence is measured by the number of bits of accuracy of the approximate point to the actual optimal point, so the iteration resembles Newton's method.

The major constraint of the inner loop is that the new approximation is an interior point. This constrains how long a step length can be taken. The solution is to *reformulate* the approximation problem for each inner loop to a canonical approximation problem using affine or projective transformations. The inverse transformation is used to map the new approximation in the reformulated problem back to a new approximation in the original problem space. Basically, the canonical approximation problem assumes that the interior point is at location $(1, 1, 1, 1...1)^T$, and the problem is to find a better point whose co-ordinates are non-negative, i.e. which is in the positive orthant. The solution is simple - take a unit step length in the optimal direction. This basic idea is modified to define a polyhedra as the intersection of the positive orthant and an affine subspace. In this case the old approximation and the new approximation are projected onto the affine subspace. An affine subspace is defined by the following parametric equation:

$Ax = b$



Linear Optimization Algorithms



Reformulation in Inner Loop of Modified Karmarker Algorithm to Canonical Problem

Thus the skeletal algorithm is an iterative approximation with the inner loop requiring a transformation to a canonical approximation problem, solving the canonical approximation problem (using a linear least squares approximation for projecting onto the affine subspace), and then transforming the new approximation back onto the original problem space.

4 Karmarker Algorithm Derivation

The Karmarker polynomial time algorithm for linear programming has sparked tremendous interest in the operations research community and has inspired others to investigate similar methods. The first theoretically polynomial time algorithm for linear optimization [Kha79] was not a practical alternative to the simplex algorithm. In practice the simplex algorithm is linear in the number of constraints, though in the worst case can be exponential in the number of variables. The Karmarker algorithm and some variants are both theoretically polynomial and are competitive with the simplex algorithm in practice. These iterative algorithms are based upon repeated projective transformations to a canonical problem of optimization over a sphere inscribed in the feasible region. It is this reformulation to a canonical problem which is of major interest with respect to my reformulation methods.

In this section a simple variant based upon repeated affine transformations is derived. This algorithm has the same skeleton as the original Karmarker algorithm, and appears to have substantial practical value. Interestingly, this algorithm is not theoretically polynomial time. This is because affine geometry is not as abstract as projective geometry (affine geometry preserves parallelism), and hence projective transformations are more powerful and can lead to theoretically faster convergence. Despite being theoretically less powerful, this simple variant illustrates the abstraction/implementation method of reformulation.

The key to the abstraction/implementation method of reformulation, as applied to geometrical problems, is to find the transformations which leave the problem invariant. This information can then be used in two ways. In the first method, called canonicalization which is illustrated here, these transformations are used to reformulate a problem to a canonical problem. For example, in a combinatorial problem involving lists where the problem is invariant under reordering of the lists, then the lists could be sorted to make the solution more efficient. In the second method, which is developed in [Low87], the definition of the problem is abstracted into terms which are invariant under the transformations. For example, lists which can be reordered would be abstracted to bags. This method is more powerful than the first since *any* implementation of bags can be chosen, not just canonicalized lists.

The formal specification of the problem from which an algorithm will be derived is the standard form for linear optimization[2]:
Minimize cx
such that $Ax = b$ and $x_1, \ldots, x_n \geq 0$

The first step in the invariant method is to symbolically solve for transformations which leave the problem invariant. An alternative to carrying out symbolic matrix algebra is *Invariant Logic* [Low87]. Invariant Logic solves for problem invariants using the invariants of known relations:

$Invariant(Incidence, ProjectiveTransformations)$
$Invariant(Ordering, OrientedProjectiveTrans)$
$Invariant(NonNegative, PositiveScaling)$[3]

Groups of invertible transformations (over an N dimensional vector space defined on a Field F) form a lattice under inclusion. The bottom of the lattice is the singleton group consisting of the identity transformation. The top of the lattice is all possible invertible transformations. A problem is guaranteed to be invariant under the intersection of the invariants of the relations which define the problem. This intersection can be found through the join of the invariants of the composing relations, which can be thought of as the greatest common denominator. This simple method will not always find the *most* general group of transformations which leave a problem invariant, for a more powerful method see [Low87].

The simplified method of Invariant Logic is to classify the constituent parts of the problem definition by their invariants and then to intersect these invariants. For example, $Ax = b$ is an incidence relation and hence invariant under all projective transformations. $Minimize\ cx$ is an ordering relation and an incidence relation, so it is invariant under oriented-projective-transformations. After applying Invariant Logic to this definition of linear optimization through simple forward chaining, the following theorem is derived:

$Invariant(LinearOptimization, PositiveScaling)$[4]

The method of parameterized theory instantiation is illustrated by instantiating the schema for iterative im-

[2] Some authors call this the canonical form

[3] Although positive translations also leave non-negativity invariant, their inverses, negative translations, do not leave non-negativity invariant. Actually, non-negative preserving invertible transformations are more general than positive scaling transformations. For example, Karmarker used a projective transformation that preserved non-negativity.

[4] Only in the canonicalization method does non-negativity need to be kept invariant. In the more general abstraction/implementation method, the non-negativity constraint is abstracted to the intersection of half-spaces.

provement for linear optimization. This forms the top-level skeleton of the Karmarker algorithm. The first step is to instantiate the abstract input-output relation to the actual input-output relation. This involves generating a substitution for the parameters *D,S,out,and Cost-Relation* such that the instantiated axioms are provable. For iterative improvement, the axioms of the abstract input-output relation are that the cost-relation between feasible outputs is total and transitive.

Iterative Improvement Schema[5]

Abstract Input-Output Relation:
Domain D
Cost-Relation: DxD
Input S $\subseteq$ D
Output out $\in$ S
IORelation $\forall y \in S\ CostRelation(out, y)$
Axioms
$\forall x, y \in D\ CostRelation(x, y)\ OR\ CostRelation(y, x)$
$\forall x, y, z \in D\ CostRelation(x, y)\ AND\ CostRelation(y, z) \Rightarrow CostRelation(x, z)$
Additional Problem Structure:
Better-element: D $\rightarrow$ D
Axioms for additoional problem structure:
$\forall x \in D\ BetterElement(x) \in D$
$\forall x \in D\ CostRelation(x, BetterElement(x))$

The following instantiation of the abstract input-output relation successfully matches the formal specification for linear optimization:
$D \mapsto R^N$
$S \mapsto \{x \mid Ax = b\ AND\ x_i \geq 0\}$
$CostRelation(x, y) \mapsto cx \geq cy$

The next step is to instantiate the additional problem structure which forms the applicability conditions of iterative improvement for optimization problems. From an AI viewpoint, this is a search problem which uses constraint propagation. In particular, the instantiation of the abstract input-output relation partially constrains the choice of the better-element function. An additional source of constraint is efficiency criteria, which are discussed in [Low87].

For this example of instantiating iterative improvement, the additional problem structure which is partially constrained is the better-element function. The constraints on this function are that its output be a member of S, and that the output has better cost than the input, as expressed in the two axioms. It is the first constraint, that the output be a member of S, which generates the canonicalization reformulation.

[5]Newton's Method, among others, is a specialization of this schema.

Using the following domain knowledge about linear least squares, the constraint that the output of Better-element be in the affine subspace defined by $Ax = b$ is factored out, leaving the constraint of non-negativity (Linear Least Squares is abbreviated LLS, while Affine Spaced is abbreviated AfS):
$\forall x, AfS\ LLS(x, AfS) \in AfS$
$\forall x, y, AfS\ LLS(x + y, AfS) = LLS(x, AfS) + LLS(y, AfS)$
$\forall x, y, c, AfS\ cx \geq cy \Rightarrow c(LLS(x, AfS)) \geq c(LLS(y, AfS))$

These equations are elementary consequences of linear least squares being an orientation-preserving linear operator. A simple consequence of these equations is the following theorem which is used in turn to derive a sufficient condition on the better-element function.

$\forall x, y, c, AfS\ cy \geq 0 \Rightarrow c(LLS(x + y, AfS)) \geq c(LLS(x, AfS))$

$\forall x, y, c, AfS\ x_i \geq 0\ AND\ cy \geq 0\ AND\ x + y \geq 0 \Rightarrow BetterElement(LLS(x, Ax = b)) = LLS(x + y, Ax = b)$

Thus constraint propagation yields sufficient conditions on an increment vector y for iterative improvement. These conditions are independent of the particular affine subspace, because the linear least squares operator incorporates the constraint of being in the affine subspace. It is these sufficient conditions which can be reformulated through positive-scaling transformations to the canonical problem of optimization over a unit sphere centered at the vector $(1, 1, 1, ..1)^T$. We assume that the following domain knowledge is available to STRATA:

Inscribed(Sphere((1,1,1,1...1),1),positiveorthant)

Optimize(direction,sphere) = *center + radius * direction*

Through additional instantiation, the vector x is mapped to the center of the unit sphere and the increment vector y is mapped to the radius*direction. This gives a full instantiation of the BetterElement function for any direction vector and affinespace *if* the current interior point is the unit vector $(1, 1, 1, ..1)^T$. However, Invariant Logic has shown that the linear optimization problem is invariant under positive scaling! Thus, it is always possible to reformulate to the canonical problem where the current interior point is the unit vector. This yields the algorithm discussed in section 3.

The hand simulated derivation given above captures the major features of the Karmarker linear programming algorithm. The methods of Invariant Logic and Parameterized Theory Instantiation play a critical role in incorporating domain knowledge and algorithm knowledge into the derivation. This derivation ignores many special cases, which would be tedious to present

in this paper. At present, my research work includes mechanizing this derivation and extending the methods to incorporate efficiency constraints.

5 Summary

This paper has discussed the capabilities needed for deriving Image Understanding algorithms. Automatically formulating an ill-structured problem as a formal mathematical problem is beyond the horizon of current AI research. Deriving an efficient algorithm for a formal mathematical problem is the subject of this paper.

One major requirement is to represent high-level programming knowledge in a clean modular representation. This facilitates the interaction of domain knowledge and programming knowledge. Parameterized theories, such as the theory for iterative improvement, are used in STRATA to represent high-level programming knowledge. Instantiating the parameters for a particular problem, such as linear optimization, results in a top-level description of an algorithm. Since the parameters are mutually constraining through the axioms of the theory, instantiating the parameters is a constraint satisfaction problem.

The second major requirement for deriving geometric algorithms is to extend abstract data types to the geometric domain. From the viewpoint of modern geometry, finding an abstract description of a geometric problem is equivalent to finding the group of transformations which leave it invariant. This paper describes *Invariant Logic* for determining the invariants of a problem. The invariants can be used to reformulate a problem to a known canonical problem.

The STRATA automatic programming system uses parameterized theory instantiation and invariant logic to derive algorithms for formal geometric problems. These two methods are illustrated with the derivation of a variant of the Karmarker linear programming algorithm. These methods are generalizeable to the derivation of other algorithms in Image Understanding.

6 Acknowledgements

This paper benefited from discussions with Professor Thomas Binford, Dr. Yinyu Ye (Stanford–Operations Research), Dr. Joseph Goguen (SRI–Theory of Abstract Data Types), Dr. George Stolfi (Stanford–Computational Geometry), Dr. Douglas Smith and Dr. Cordell Green (Kestrel Institute–knowledge based automatic programming). This work was supported in part by DARPA contract N00039-84-C-0211, and ONR contract N00014-84-C-0473. The views and conclusions contained in this paper are solely those of the author and should not be interpreted as representing the official policies, either expressed or implied of the U.S. Government.

References

[Bar86] David Barstow. An experiment in knowledge-based automatic programming. In Charles Rich and Richard C. Waters, editors, *Artificial Intelligence and Software Engineering*, Morgan Kaufmann Publishers, Inc., 1986.

[CK85] Dong J. Choi and John R. Kender. Solving the depth interpolation problem with the adaptive chebyshev acceleration method on a parallel computer. In *Proceedings: Image Understanding Workshop*, pages 219–223, DECEMBER 1985.

[Fea82] Martin M Fischler and et al. Modeling and using physical constraints in scene analysis. In *AAAI-82*, August 1982.

[GB85] Joseph A. Goguen and Richard M. Burstall. *Institutions: Abstract Model Theory for Computer Science.* Technical Report CSLI–85–30, CSLI, 1985.

[GTW78] Joseph Goguen, Jim Thatcher, and Eric Wagner. An initial algebra approach to the specification, correctness, and implementation of abstract data types. In Raymond T. Yeh, editor, *Current Trends in Programming Methodology*, pages 80–149, Prentice-Hall, 1978.

[Kha79] L. G. Khachiyan. A polynomial algorithm for linear programming. *Soviet Math Doklady*, 20:191–94, 1979.

[KN83] Elaine Kant and Allen Newell. An automatic algorithm designer: an initial implementation. In *AAAI-83*, 1983.

[Lee85] David Lee. A provably convergent algorithm for shape from shading. In *Proceedings: Image Understanding Workshop*, pages 489–495, DECEMBER 1985.

[LM81] Michael R. Lowry and Allen Miller. A general purpose vlsi chip for computer vision with fault-tolerant hardware. In *Image Understanding Workshop*, April 1981.

[LM82] Michael R. Lowry and Allen Miller. Analysis of low-level computer vision algorithms for implementation on a vlsi processor array. In *SPIE-82*, August 1982.

[Low82] Michael R. Lowry. Reasoning between structure and function. In *Image Understanding Workshop*, September 1982.

[Low87] Micheal R. Lowry. *Algorithm Synthesis through Problem Reformulation*. PhD thesis, Stanford University, 1987.

[PRH85] Igor Pavlin, Edward Riseman, and Allen Hanson. Analysis of an algorithm for detection of translational motion. In *Proceedings: Image Understanding Workshop*, pages 388-399, DECEMBER 1985.

[Ric86] Charles Rich. A formal representation for plans in the programmer's apprentice. In Charles Rich and Richard C. Waters, editors, *Artificial Intelligence and Software Engineering*, Morgan Kaufmann Publishers, Inc., 1986.

[Smi85] Douglas R. Smith. Top-down synthesis of divide-and-conquer algorithms. *Artificial Intelligence*, 27(1), September 1985.

[SSS86] Edmond Schonberg, Jacob T. Schwartz, and Micha Sharir. An automatic technique for selection of data representations in setl programs. In Charles Rich and Richard C. Waters, editors, *Artificial Intelligence and Software Engineering*, Morgan Kaufmann Publishers, Inc., 1986.

[WBKL83] Patrick Winston, Thomas Binford, Boris Katz, and Michael R. Lowry. Learning physical descriptions from functional definitions, examples, and precedents. In *AAAI-83*, August 1983.

GENERALIZING EPIPOLAR-PLANE IMAGE ANALYSIS FOR NON-ORTHOGONAL AND VARYING VIEW DIRECTIONS *

H. Harlyn Baker
Robert C. Bolles
David H. Marimont

Artificial Intelligence Center
SRI International
333 Ravenswood Avenue
Menlo Park, CA 94025.

Abstract

Earlier reports have described the theory and practice of Epipolar-Plane Image Analysis. Our previous implementations of this mapping technique demonstrated the feasibility and benefits of the approach, but were carried out for restricted camera geometries. The question of more general geometries made utility for autonomous navigation uncertain. Here, we discuss a generalization of the analysis that a) enables varying view direction (including varying over time), b) provides three-dimensional connectivity information for building coherent spatial descriptions of observed objects, and c) raises the possibility of removing the restriction of travel along a linear trajectory.

1: Introduction

Approaches to depth measurement through stereo analysis suffer from the dichotomy of choosing between a wide baseline, with high precision of matched features but both increased match failures and increased difficulties of perspective and occlusion effects, and a narrow baseline, with easy matching but poor depth accuracy. This is an obvious limitation confronted whenever depth measurements must be made on the basis of just 2 views of a scene. In this work we take a direction that gains the advantages of both approaches. We process a large number of closely spaced images: the close spacing makes matching easy, and the large number of images means a wider baseline, and therefore higher accuracy. While bearing the increased cost of processing the large number of images, and having to know precisely the position and attitude of the camera at each imaging site, our technique brings significant advantages in accuracy and reliability over existing depth measurement approaches.

Our approach (see [2] and [3] for details) is to take a sequence of images from positions that are very close together – close enough that almost nothing changes from one image to the next. With this capture spacing, none of the image features moves more than a few pixels. This sampling frequency guarantees a continuity in the temporal domain similar to the obvious spatial continuity, and lets us avoid the difficult correspondence problem. An edge of an object in one image appears temporally adjacent to (within a pixel or so of) its occurrence in both the preceding and following images. This rapid sampling makes it possible to construct a volume of data (spatio-temporal data) in which time is the third dimension and continuity is maintained over all three dimensions (Figure 1). In one of the traditional motion-analysis paradigms features are detected in successive spatial images and matched from one to the next, with the difference in position used to deduce motion (see, for example Roach [11]). In another, differential techniques are used between successive images to measure the direction of intensity variation, and thereby the direction of inferred motion (for example, Buxton [6] and Waxman [12]). We, however, take an approach orthogonal to these (literally). We slice the spatio-temporal data along a temporal dimension, locate features in these slices, observe their motion over time, and use this motion to determine the three-dimensional feature locations. This temporal slicing, into what we call *epipolar-plane images*, or EPIs, can be done whenever the camera moves in a straight line (the direction and complexity of the slicing varies with the camera parameters). We discuss the geometrical considerations of this approach, point out its major strengths and weaknesses, and present the mechanism for generalizing the analysis to an unrestricted range of camera viewing directions along a linear, and possibly non-linear, path.

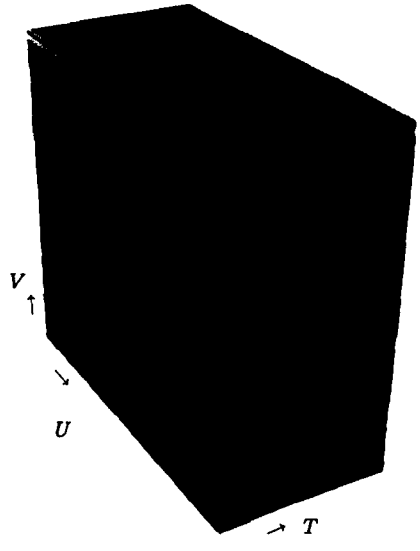


Fig. 1. Spatio-temporal volume of data.

*This research was supported by DARPA Contracts MDA 903-86-C-0084 and DACA 76-85-C-0004.

2: Analysis in the Epipolar Plane

We model the camera as a pin-hole with image plane in front of the lens (Figure 2). For each feature P in the scene and two viewing positions V_1 and V_2, there is an *epipolar plane*, which passes through P and the line joining the two lens centers. This plane intersects[1] the two image planes along corresponding *epipolar lines*. An *epipole* is the intersection of an image plane with the line joining the lens centers. In motion analysis, an epipole is often referred to as the focus of expansion (FOE) because the epipolar lines radiate from it. In our work, the camera moves in a straight line, and the lens centers at the various viewing positions lie along this line. Here, the FOE is the camera path. This structuring divides the scene into a pencil of planes passing through the camera path. We view this as a cylindrical coordinate system with axis (H) the camera path, angle (θ) defined by the epipolar plane, and radius (R) the distance of the feature from the axis. Note that a scene feature is restricted to a single epipolar plane, and any scene features at the same angle (within the discretization) share that epipolar plane. This means that the analysis of a scene can be partitioned into a set of analyses, one for each EPI, and these EPIs can be processed independently and in parallel.

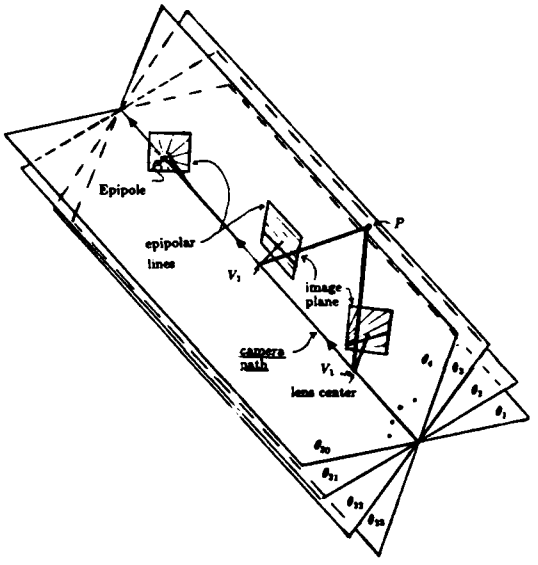


Fig. 2. General epipolar configuration.

Figure 3 shows a simple motion with a camera moving orthogonal to its direction of view. Here, the epipolar lines for a feature, such as P, are horizontal scanlines, and these occur at the same vertical position (scanline) in all the images. Each scanline is a projected[1] observation of the features in the epipolar plane. The projection of P onto these epipolar lines moves to the right as the camera moves to the left. For a constant camera motion, the velocity of this movement along the epipolar line is a function of P's distance from the line joining the lens centers. The closer the feature, the greater is its motion. For this type of lateral motion the trajectories are straight lines (see Adelson [1], Bridwell [4] and Yamamoto [13] for some simple observations on this structuring that predate our use of EPIs). Figure 4 shows a set of epipolar lines (here, scanlines) made into an image (an EPI). Each horizontal line of the EPI is one image scanline. Time progresses from bottom to top, and, as the camera moves to the left, the features move to the right.

[1]Here, intersection and projection are equivalent.

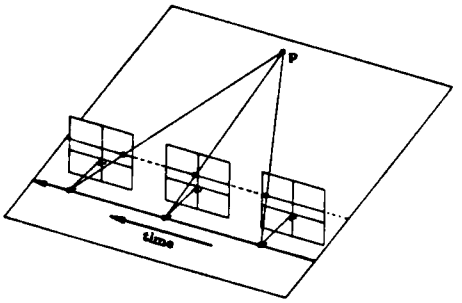


Fig. 3. Right-to-left motion.

Fig. 4. Epipolar-plane image from right-to-left motion.

There are several things to notice about this image. First, it contains only linear structures. Second, the slopes of the lines determine the distances to the corresponding features in the world – the greater the slope, the farther the feature. Third, occlusion, which occurs when a closer feature moves in front of a more distant one, is immediately apparent in this representation. These suggest the approach we should take in analyzing the data: find the linear structures (feature paths) in the EPIs; join together paths that are collinear, although broken by occlusion; use the slopes of the linear paths to determine scene depth.

The EPI in Figure 4 was constructed from a simple right-to-left motion with the camera oriented at right angles to the path. As long as the lens center of the camera moves in a straight line the epipolar planes remain fixed relative to the scene (as in Figure 2). The camera can even change its orientation about its lens center as it moves along the line without affecting this partitioning of the scene. Orientation changes move the epipolar lines around in the image plane, significantly complicating[2] the construction of the EPIs, but the epipolar planes remain fixed. Figure 5a is an EPI formed from a sequence of images taken by a camera looking straight ahead as it moves – notice that the feature paths are hyperbolic. Allowing the camera to change orientation smoothly as it moves along gives rise to EPIs with curves as those in Figure 5b.

These non-orthogonal viewing directions would seem to make our tracking problem non-linear. To maintain the linearity regardless of viewing direction, we chose a representation that leads to an approach different from that outlined above. We do not find linear feature paths in the EPIs; rather, we find linear paths in a dual space. Our insight here (see Marimont [8] and [9]) is that no matter where a camera roams about a scene, for any particular feature, the lines of sight from the camera principal point through that feature in space (determined by the line from the principal point through the point in the image plane where the projected feature is observed) all intersect at the feature (ignoring measurement error). The duals of these lines of sight lie along a line whose dual is the scene point: fitting a point to the lines of sight is a linear problem.

[2]Notice that only in the case of orthogonal viewing will the intersections of epipolar planes with image planes be parallel and feature paths be straight lines.

Fig. 5a. EPI with forward view direction.

Fig. 5b. EPI with varying view direction.

Capitalizing on these observations, our analysis of feature motion in the epipolar planes consisted of the following steps:

1. 3D convolution of the spatio-temporal data
2. Slicing the convolved data into EPIs
3. Detecting edges in the EPIs and converting them to lines of sight
4. Breaking these lines of sight into linear segments (in Dual space)
5. Merging collinear segments
6. Computing x-y-z coordinates
7. Building a map of free space
8. Linking x-y-z points between EPIs

Our first implementation was described in Bolles [2], and the current (as above) implementation is discussed and demonstrated in Bolles [3].

The following should be noted about this analysis:

- There are some obvious ways to make it incremental in time, and partitionable in V (epipolar planes), for high speed performance;
- Although, in principle, the camera's view direction is unrestricted, its motion must be in a straight line[3];
- Frame rate must be high enough to limit the frame-to-frame changes to a pixel or so (more specifically, such that the motion of a surface is less than its projected width);
- Independently moving objects will either not be detected, or will be detected inaccurately;
- Curved objects may either be viewed as polygonal approximations, or appear as moving objects (see [3]).

3: Generalization

Although demonstrating the feasibility of this novel approach to depth analysis, the implementation described above had several significant limitations:

1. While the theory placed no restriction on camera view direction, the implementation could not handle any motions other than those along a linear path where the view direction was *fixed* and *perpendicular* to the motion. It could not handle the obvious cases of view in the direction of motion, or view changing with motion.
2. *Working with epipolar planes separately*, there was no satisfactory way to combine results between planes, or, of equal importance, to share information between nearby planes in producing the results. This led to fragmentation of the depth results, and a lacking in three-dimensional coherence[4] for both depth results and free space (see [3]).
3. The processing was structured to work on an entire image sequence at one time, requiring that all images be taken before analysis was begun.
4. There was no way, within the framework developed, to remove the restriction to a linear camera path.

These restrictions put a severe lower limit on the ability of the technique to show any practical usefulness for real tasks of navigation or 3-D sensing. We have been looking at these limitations, and developing new techniques to remove them. In fact, the mechanisms for their removal all rest on a single restructuring of the analysis.

We collect the data as a sequence of images. As each new image is acquired, we construct its *spatial* and *temporal* edge contours. These contours are three-dimensional zeros of the Laplacian of a chosen three-dimensional Gaussian[5] (LoG), and the construction produces a spatio-temporal *surface* enveloping the positive (or negative) *volumes* (in two-dimensions, edge contours envelop positive or negative *regions*). The *spatial* connectivity in this structure lets us explicitly maintain object coherence between features observed on separate epipolar planes (the second limitation above); the *temporal* connectivity gives us, as before, the tracking of features over time, and therefore, for known camera parameters, the positions of those features in space. We can use the spatial connectivity *after* processing, to form connected feature descriptions, and also *during* the analysis, in giving us greater support for determining our estimates (incorporating weighted observations from adjacent epipolar planes) and letting us associate features that appear disjoint in the epipolar plane in which they occur, but lie on a common surface (*ie*, occlusion is not complete over all time and space, and there is some perspective from which the features are not disjoint).

In this spatio-temporal surface description, feature observations bear (u, v, t) coordinates, and are spatio-temporal *voxel facets*. Figure 7 shows a mesh description of the facets for the spatio-temporal surface associated with the forward-viewing sequence whose first and last images[6] are depicted in Figure 6 – surfaces are interpolated frontiers separating the positive and negative LoG voxels. Now, for non-orthogonal camera viewing directions, epipolar lines are not scanlines (i.e. the epipolar planes are not distinguished by the v coordinate). To obtain this necessary structuring, we develop within this spatio-temporal description an embedded representation that makes the epipolar

[3] If the lens center does not move in a straight line, the epipolar planes passing through a world point differ from one camera position to the next. The points in the scene are grouped one way for the first and second camera positions, a different way for the second and third, and so on. This makes it impossible to partition the scene into a fixed set of planes, which in turn means that it is not possible to construct EPIs for such a motion. The arrangement of epipolar lines between images must be transitive for EPIs to be formed.

[4] We aren't alone in this – no other depth sensing technique produces this explicit coherence either.

[5] See also Buxton [5] and Heeger [7] for their use of spatio-temporal convolution over an image sequence.

[6] The surface representations shown here are based, for clarity and development, on a reduced version of the imagery – one eighth the linear resolution of the originals.

Fig. 6. First and last images of forward viewing sequence.

Fig. 7. Spatio-temporal surface representation for the first 10 frames.
(crossed-eye display, one eighth linear resolution)

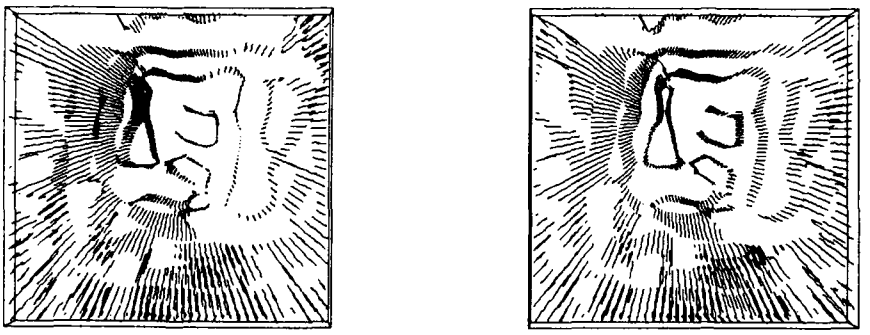


Fig. 8. Epipolar-plane surface representation.

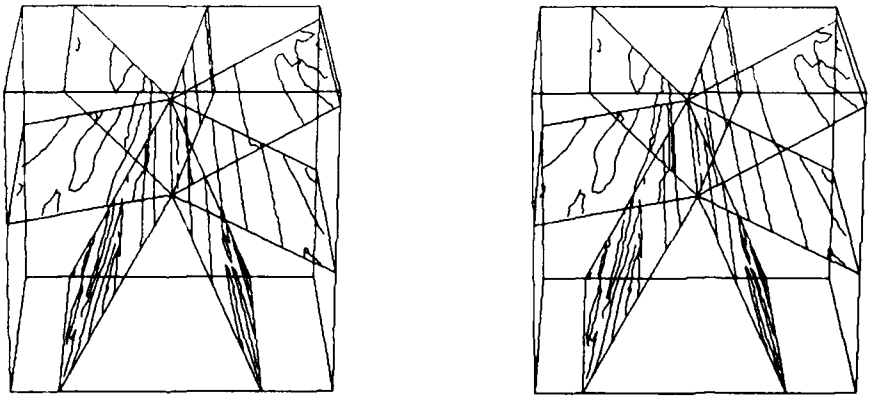


Fig. 9. Intersection of several epipolar planes with spatio-temporal surfaces.
(30 frames)

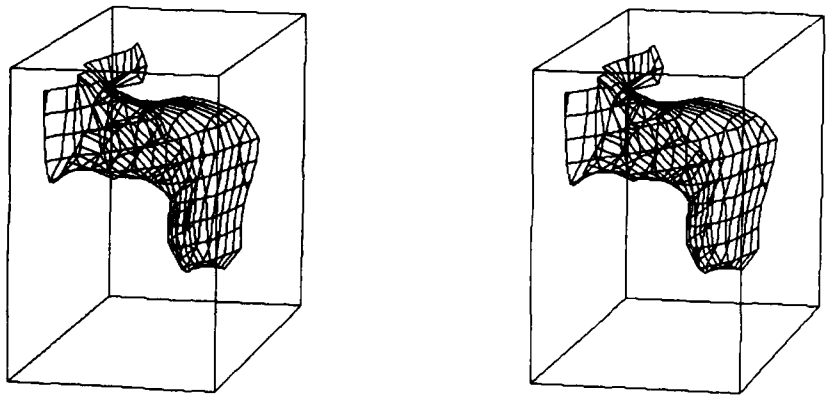


Fig. 10. Single spatio-temporal surface from top left of Figure 7.

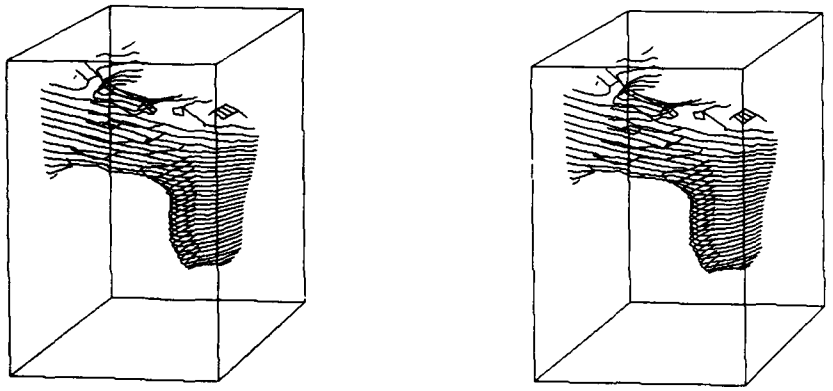


Fig. 11. Epipolar-plane representation for surface of Figure 10.

organization explicit. Over each of the sequential images, we transform the (u, v, t) coordinates of our spatio-temporal zeros to (r, h, θ) cylindrical coordinates (θ indicates the epipolar-plane angle ($\theta \in [0, 2\pi]$), the quantized resolution in θ is a supplied parameter, and the transform for each image is determined by the particular camera parameters). In this new coordinate system, we build a structure very much like our earlier epipolar-plane edge contours, but organized by epipolar plane. This is done by intersecting the spatio-temporal surfaces with the pencil of appropriate epipolar planes[7] (as Figure 2). We weave the epipolar connectivity through the spatio-temporal volume, following the known camera viewing direction changes. Figure 8 shows a sampling of the spatio-temporal surfaces as they intersect the pencil of epipolar planes (every fifth plane is depicted). Figure 9 shows some of these surface-plane intersections, along with the associated bounding planes (refer to Figure 2). Figure 10 isolates a single surface from the top left of Figure 7, and shows it's spatio-temporal structure. Figure 11 shows the same surface structured by its epipolar-plane components.

4: Assessment

By maintaining the epipolar structure irrespective of camera attitude, we have provided a remedy to the first limitation cited above (non-orthogonal and varying viewing directions). In these planes, feature paths will be, in general, arbitrary curves in (u, v, t) space, but lines in the space of lines of sight.

[7]Notice that, with view direction varying, the underlying epipolar plane may be far from planar in (u, v, t) space – it may undulate in a manner similar to that in which Figure 5b varies from Figure 4, and for similar reasons.

The fact that the analysis is performed as images are acquired means that the processing may be incremental in time, and this removes the requirement mentioned in limitation 3 where, earlier, the entire sequence had to be obtained before processing could be begun. The epipolar re-structuring is also incremental in time. We are still developing the incremental feature estimation.

A crucial constraint of the current epipolar-plane image analysis is that having a camera moving along a linear path enables us to divide the analysis into planes, in fact, the pencil of planes that passes through the camera path. With this, we are assured that a feature will be viewed in just a single one of these planes, and its motion over time will be confined to that plane. Another crucial constraint is the one we generalized from the orthogonal viewing case – we know that the set of line-of-sight vectors from camera to feature over time will all intersect at that feature, and determining that feature's position is a linear problem. This latter constraint does not depend upon the former. In fact, the problem would remain linear even if the camera meandered all over the scene. This knowledge gives us a possibility of resolving the fourth limitation above – that the camera path be linear. All that the linear path guarantees is that the problem is divisible into epipolar planes. If we lose this constraint, then we cannot restrict our feature tracking to separate planes. The features will, however, still form linear paths in the space of *line-of-sight vectors*, and our spatio-temporal surface description is an appropriate representation for doing this non-planar, but still linear, tracking. The motion of features will give us *ruled*

surfaces, with the rules (zeros of gaussian curvature) revealing the positions of the features in space[8]. This generality suggests that there is even broader application for the technique than we had initially thought.[9]

5: Conclusions

We showed, in our earlier work, the feasibility of extracting scene depth information through *Epipolar-Plane Image Analysis*. In this, our results in determining scene depth are distinctive in that they:

- bridge the dichotomy between accuracy and ease of matching;
- through the use of an increasing baseline, give highly accurate estimates of feature position, carrying associated measures of that accuracy for each feature tracked;
- both indicate occluding contours, and use them in collecting observations to improve feature estimates;
- provide not only feature 3-D position measures, but also information about scene free space – information which is traditionally quite difficult to acquire;

Our theory applies for any motion where the camera (lens center) moves in a straight line, and the reported implementations covered the case of viewing direction orthogonal to the camera path. The generalizations obtained through spatio-temporal surface analysis, demonstrated here and currently being refined, bring us the advantages of:

- incremental analysis;
- unrestricted viewing direction (including direction varying along the path);
- spatial coherence in our results, providing connected surface information for scene objects, rather than point estimates structured by epipolar plane;
- suggesting it may be possible to remove the restriction that fixes us to a linear path.

[8] Visualize pick-up-sticks jammed in a box, with the sticks being the rules.

[9] Although being computationally expensive, it would be possible here to use the pairwise epipolar constraints between images to constrain rule tracking on the spatio-temporal surface.

References

[1] Adelson, Edward H., and James R. Bergen,. "Spatiotemporal energy models for the perception of motion," *Journal of the Optical Society of America A*, 2:2 (1985), 284–299.

[2] Bolles, R. C., and H. H. Baker, "Epipolar-Plane Image Analysis: A Technique for Analyzing Motion Sequences," *Proceedings: DARPA Image Understanding Workshop*, Miami Beach, Florida, (December 1985) 137–148.

[3] Bolles, Robert C., H. Harlyn Baker, and David H. Marimont, "Epipolar-Plane Image Analysis: An Approach to Determining Structure from Motion," premier issue of *The International Journal of Computer Vision*, Kluwer Academic Publishers, to appear, January 1987.

[4] Bridwell, Nelson J., and Thomas S. Huang, "A Discrete Spatial Representation for Lateral Motion Stereo," *Computer Vision, Graphics, and Image Processing*, 21 (1983), 33–57.

[5] Buxton, B.F., and Hilary Buxton, "Monocular Depth Perception from Optical Flow by Space Time Signal Processing," *Proceedings of the Royal Society of London Series B* 218 (1983), 27–47.

[6] Buxton, B.F., and H. Buxton, "Computation of Optic Flow from the Motion of Edge Features in Image Sequences," *Image and Vision Computing*, 2:2 (1984).

[7] Heeger, David J., "Depth and Flow from Motion Energy," *Proceedings of the Fifth National Conference on Artificial Intelligence (AAAI-86)*, (Philadelphia, August 1986), 657–663.

[8] Marimont, David H., "Inferring Spatial Structure from Feature Correspondences," Ph.D. dissertation, Stanford University, 1986.

[9] Marimont, David H., "Projective Duality and the Analysis of Image Sequences," *Proceedings of the Workshop on Motion: Representation and Analysis*, IEEE Computer Society, Kiawah Island, South Carolina, (May 1986), 7–14.

[10] Moravec, Hans P., "Visual Mapping by a Robot Rover," *Proceedings of the International Joint Conference on Artificial Intelligence*, (Tokyo, August 1979), 598–600.

[11] Roach, J.W., and J.K. Aggarwal, "Determining the movement of objects from a sequence of images," *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-2:6 (1980), 554–562.

[12] Waxman, Allen M., and Kwangyoen Wohn, "Contour Evolution, Neighborhood Deformation, and Global Image Flow: Planar Surfaces in Motion," *The International Journal of Robotics Research* 4:3 (1985), 95–108.

[13] Yamamoto, M., "Motion Analysis Using the Visualized Locus Method," untranslated Japanese articles, 1981.

Detecting Dotted Lines and Curves in Random-Dot Patterns*

Richard Vistnes

AI Lab, Stanford University, Stanford, California 94305

Abstract

Results are reported of psychophysical experiments that measured human performance in detecting dotted lines and curves embedded in a random-dot background. As pattern density increases, more point scatter and curvature is tolerated. A statistical model for the detection of non-accidental patterns such as lines and curves is presented; the model is based on center-surround operators that find significant differences in dot density in an elongated region as compared to the local surround. Performance predicted by the model compares well with experimental results, as do results of computer simulations.

1 Introduction and Motivation

This paper examines one aspect of human perceptual organization of images: the preattentive grouping of image elements into lines and curves. I report the results of a series of experiments in which subjects tried to detect dotted line segments and dotted circular arcs surrounded by randomly-placed, identical dots. I develop a theory of dotted line and curve detection, from which predictions are made and compared to the experimental data. A computer simulation, and the characteristics of its performance, offers another perspective on the mechanisms underlying the perceptual phenomenon.

The motivations for performing these experiments are twofold. First, current notions of edge detection in computer vision systems involve detecting local discontinuities in the image, followed by linking these "edgels" into smooth curves [1,11]. Zucker [19,22,23] has studied the linking of edgels using only the geometrical information in the edgel positions. This removes information about contrast, orientation, size and other factors, and simplifies the problem; I follow the same approach here. One motivation for studying human performance is to discover new algorithms that can be used in computer vision systems.

Another motivation is to clarify the nature of human preattentive processing. The grouping problem studied here deserves attention, since humans can easily recognize objects depicted in briefly-presented line drawings [2]. My own informal demonstrations show that people can recognize briefly-presented dotted-line drawings of objects as well, even when those dotted lines are embedded in a noisy background. Since only preattentive processes are at work in these recognition tasks, we conclude that the information obtained from them is sufficient for recognition, at least for the class of shapes used in the experiments. This paper concentrates on aspects of preattentive line and curve detection. Others, notably Julesz [7] and Treisman [13], have studied other aspects of preattentive perception.

1.1 Definitions

How shall we define a dotted line? Restricting the definition to *straight* lines of dots is pointless, since a slightly jagged dotted line does not satisfy this definition, yet we still perceive it as a line. Consider the dotted lines shown in Fig. 1. We certainly consider the structure at the top to be a line, and probably the one in the middle too. But what about the one at the bottom, with its distantly-spaced dots? It is not clear whether we should call this structure a line or not; that decision must depend on the context and on the goals of the vision system. If this bottom structure *is* to be considered a line, it is not clear where its endpoints are, nor which dots it contains.

Thus, it can be misleading to *decide* whether or not an image structure constitutes a line. Imagine drawing the dots in the bottommost structure in Fig. 1 closer and closer together: the closer the dots, the stronger the impression of a line. There is no definite point at which we call more closely-spaced dots *a line*, but more distantly-spaced dots *not a line*. Furthermore, the significance of a pattern depends to a great extent on the nearby image. The bottom line in Fig. 1 is unnoticeable when surrounded by other dots as it is; if the dotted surround were removed or reduced in density, this line would become easily detectable.

*This is an abridged version of a paper submitted to *Journal of the Optical Society of America, Series A.*

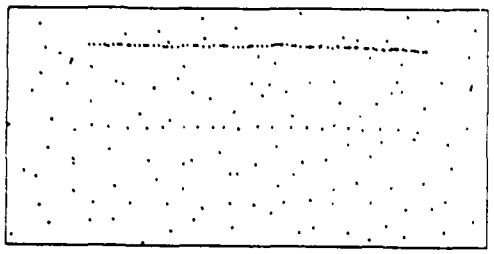


Figure 1. Dotted lines of different significance. The line at the top is very significant; the one in the middle is less significant. At the bottom the line is marginally significant.

It is more useful to assign a *significance level* to a structure. It is possible to determine the statistical significance of a pattern: how likely it is to have arisen at random, given the nature of the image nearby. If it is necessary to decide whether a structure constitutes a line or not, then the significance level can be used: e.g., if there is only a 1% chance that the structure would arise by chance, given the nearby image, then it might be useful to make that signal-to-symbol decision. We should not arbitrarily set thresholds for determining what structures we consider to be edges, lines and curves; Binford [4,3] makes the same argument. For some purposes it may be useful to ignore patterns that do not exceed some level of significance. But in general it is reasonable to define only a class of patterns for which to search in the image, and a function defined on such patterns that determines their significance.

For the purposes of this paper, it will be sufficient to define a line segment in a (dotted) image as a narrow rectangular region in which the density of dots is significantly higher than in the local background. Note that this definition of line segment treats clusters of dots the same as collinear dots. For the purpose of *detecting* line segments, however, the confusion of density with collinearity implied in the current definition should cause no problems in any real vision system. Furthermore, the implications of such a definition seem to agree with human psychophysics, as we shall see.

1.2 Preview

First, I discuss human perception of dotted lines and dotted curves in images. I review the results of my previous psychophysical experiments that determined the power and limitations of human preattentive vision in this task, and present some new results that are more amenable to analysis.

Second, I present a statistical model for what information is being computed from the image, make predictions about the limits of performance based on that model, and compare the predictions with the experimental data. I show that curves can be detected by line detectors, and that the characteristics of a mechanism that uses this approach are similar to those observed in humans. I discuss computer simulations, in which three types of line-detection operators are applied to stimuli similar to those used in the human experiments; the performance of these operators in the line detection task is presented and compared with human performance. I conclude by reviewing related research and discussing the implications of the results presented here for human and computer vision.

2 Experimental Methods

This section introduces two experiments that determined the parameters affecting the detectability of dotted lines and dotted curves. Experiment I studied dotted straight lines, Experiment II dotted curves. These target patterns were displayed to subjects for a short time (about 200 msec), preventing prolonged inspection of the image and forcing subjects to use preattentive mechanisms.

Experiment I (lines) determined how close the dots in the line need to be, compared to those in the surround, in order for the target to be detected. We expect that targets with closely-spaced dots will be easier to detect than those with dots placed farther apart; the experiments quantified this effect. Another parameter of interest is the amount of transverse variation, or point scatter, in the line. We expect that, for fixed dot separation in the target, as the transverse variation increases, the line will be more difficult to detect; again, the experiments quantify this effect.

I noticed that dotted curves are more difficult to detect than dotted lines, given equal dot separations in the target. We expect that as the curvature of a dotted arc increases, the arc will become more difficult to detect. Experiment II quantified curvature detectability limits in an arc as the dot separation varied.

The subject's task was to indicate where he or she thought the target had appeared, in a four-alternative forced-choice task. The target was located in one of four locations in the image: vertical on the left or right, or horizontal on the top or bottom. Some examples of the kind of stimuli used appear in Fig. 2. Detectability was measured by accuracy of location of the target over 50 trials.

2.1 Subjects and Apparatus

Two graduate students served as subjects. A Symbolics 3600 Lisp Machine with a black-and-white monitor generated and displayed the images. The distance from

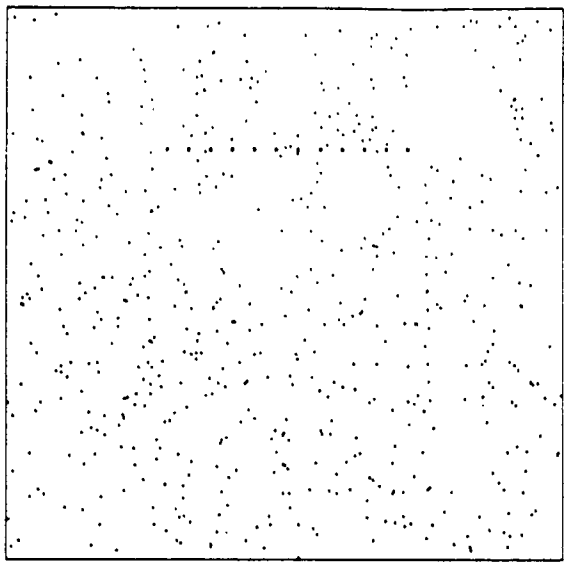


Figure 2. Examples of stimuli used in the experiments. The actual experiments contained only one target; four are shown here. Parameters: $d_s = 20$, lengths L are all 200, and $v_L = 0$. The target at top is emphasized, and has $d_t = 18$, $v_T = 0$, $H = 0$; the one at bottom is same, except $H = 40$. At left, a line with $H = 0$, $d_t = 18$, $v_T = 0.4$; at right, the same except $d_t = 15$. The actual size of the square image was 12.7 cm.

screen to subject was 55.9 cm. The size of the square display was 460 pixels (12.7 cm), subtending 13° of visual angle. The length of the targets was 200 pixels (11.0 cm), subtending 5.7°. Each dot was a 2×2 square of pixels, the side subtending 3.4 minutes of arc. The distance from the fixation point at the center of the display to the line was 6.0 cm or 3.1°.

2.2 Stimuli

The target dots are embedded in a surround of randomly-placed dots whose appearance is described by one parameter, the average spacing between dots d_s ($d$istance in $s$urround). The mean number of dots falling in the image is $A\rho$, where A is the area of the image and $\rho = 1/d_s^2$ is the dot density. An approximation to a Poisson distribution of surround dots is achieved by placing a number of dots at random in the image; the number of dots is normally distributed with mean and variance $A\rho$, as in a Poisson distribution.

Lines

The dotted-line generating procedure uses the length L of the line and the dot spacing d_t ($d$istance in $t$arget) to produce a line containing $\lceil L/d_t \rceil$ dots. A longitudinal variation v_L specifies the maximum fractional variation along the line. Dots are perturbed about their nominal (evenly-spaced) positions by $\frac{1}{2} X d_t v_L$ in either direction, where X is a uniformly-distributed random number between -1 and 1. Similarly, transverse variation v_T specifies the maximum variation in a direction normal to the line. Dots are perturbed about their nominal positions by an amount $\frac{1}{2} X d_t v_T$; X is again a random number between -1 and 1. When $v_L = 0$, the dots are evenly-spaced; when $v_T = 0$, the dots fall in a straight line. The dots generated in this manner fall in a rectangular region of length L and width $d_t v_T$. The parameters v_T and v_L provide a scale-independent way of specifying the shape of the line; magnifying or shrinking the pattern does not change v_T or v_L.

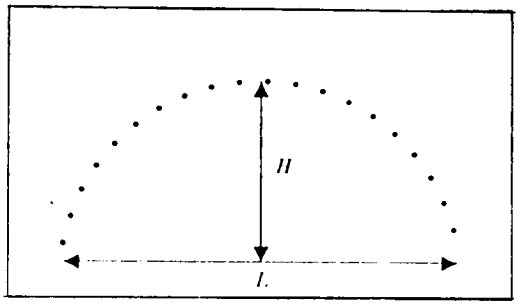


Figure 3. A dotted circular arc. Chord length is L and height is H.

Curves

The shape of an arc of evenly-spaced dots (see Fig. 3) can be specified by the length L of the chord joining the ends of the arc, the height H of the arc above this chord, and the average spacing d_t of dots in the arc. In these experiments, the arcs had no transverse or longitudinal variation. When H is zero, an arc specified in this manner reduces to a straight line as specified above. The radius of curvature of such an arc is given by

$$R = \frac{(L/2)^2 + H^2}{2H} \qquad (1)$$

and the curvature κ of the arc is just $1/R$.

2.3 Analysis of data

An experiment typically consisted of varying the value of one parameter, say v_T, while keeping all others fixed. As the parameter varied, accuracy of detection changed, usually deteriorating and approaching the chance level of 25% for the 4-alternative task. A psychometric function given by

$$P(x) = G + (\gamma - G)\exp\left[-(x/\alpha)^{\beta}\right]. \qquad (2)$$

where x is the independent variable, was fit to each such set of data points using a method described by Watson [16]. G is the expected fraction correct when the subject guesses; here, $G = 0.25$.

We are interested in the 80% accuracy threshold of performance. From (2) we obtain:

$$x = \alpha \left[\log \frac{\gamma - G}{T - G} \right]^{(1/\beta)}$$

with $T = 0.80$.

It is difficult to obtain error estimates on this threshold. However, an approximate standard deviation was obtained by using a "bootstrap" method, as follows. The fitting procedure described by Watson [16] assumes that the data points $(x, P(x))$ are Bernoulli random variables with some underlying probability $p(x)$ for the n trials. We use this underlying distribution function for each data point to generate new sets of data points; each new data point comes from the Bernoulli distribution corresponding to x. Since $p(x)$ and n completely specify the Bernoulli distribution, we find the standard deviation of observed number correct as $\sigma'(x) = \sqrt{p(x)[1 - p(x)]n}$. The standard deviation of fraction correct for each point is then $\sigma(x) = \sigma'(x)/n = \sqrt{p(x)[1 - p(x)]/n}$. A threshold value is computed for this new data set. This operation is performed several times, and the standard deviation of these thresholds is used as the standard deviation of the actual threshold.

3 Experimental Results

3.1 Previous results

A previous series of experiments, reported by Vistnes [15], used similar stimuli except that the random surrounds were more uniform in dot density. This was done by randomly perturbing the positions of a regular grid of dots. Also, dots in the surround that fell near dots in the target were erased. The dot patterns produced in this manner have no spurious clusters of dots but still appear quite random. However, the properties of such distributions are more difficult to analyze than those of a purely random (Poisson) distribution. Hence, I repeated the experiments using a purely random background.

The earlier results showed that longitudinal variation v_L had a very small effect on detectability. That is, the dots in the target did not need to be evenly spaced along the line. Target length L was also unimportant, so long as there were at least five or six dots in the line; adding dots to the line by increasing its length did not increase its detectability. To a large extent the amount of transverse variation v_T that can be tolerated in a straight line is independent of scale. That is, maximum v_T depends on the relative separation of target dots compared with surround dots, d_t/d_s. Similar results hold for curve detection: the amount of curvature that can be tolerated is largely independent of the absolute density of the patterns, but depends mainly on d_t/d_s.

3.2 Experiment I

Transverse variation v_T was the parameter of interest for straight lines. I measured how much variation could be tolerated for a given d_t and d_s, using a performance standard of 80% correct as the criterion for detectability. The results are shown in Fig. 4a. As expected, more transverse variation can be tolerated when d_t/d_s is small.

It is important to ask if these results are independent of the size and density of the image — i.e., if the mechanism is scale invariant. If it were, then since v_T is a scale-independent measure of transverse variation, the curves in Fig. 4a would coincide when plotted as a function of relative dot separation d_t/d_s, as in Fig. 4b. These curves have similar shape but do not coincide. Thus, dotted lines in dense patterns are somewhat more easily detected than in sparse patterns, other factors being equal. To a large extent, however, the important parameter for this range of dot densities is the *relative* separation of target dots compared to surround dots. Note that for nearly straight lines ($v_T \approx 0$) the dotted line can be detected essentially when the target dots are closer together than the surround dots. These results apply over a range of a factor of 2.2 in d_s, or a factor of 4.8 in dot density.

3.3 Experiment II

For curved targets, the important parameter is arc height H, or equivalently, curvature κ. The experiments measured the maximum curvature that could be detected with a fixed d_t and d_s. The results are shown in Fig. 5a. The maximum detectable curvature increases as d_t/d_s decreases.

Again, we inquire if these results are independent of scale. Fig. 5b shows the results as a function of d_t/d_s. The curves have similar shape (this time nearly linear with similar slopes), but there is again a tendency for curvature in dense patterns to be more easily detected than in sparse ones.

4 A Model for Dotted Line Detection

I now describe a statistical model that accounts in a qualitative way for these experimental data. The

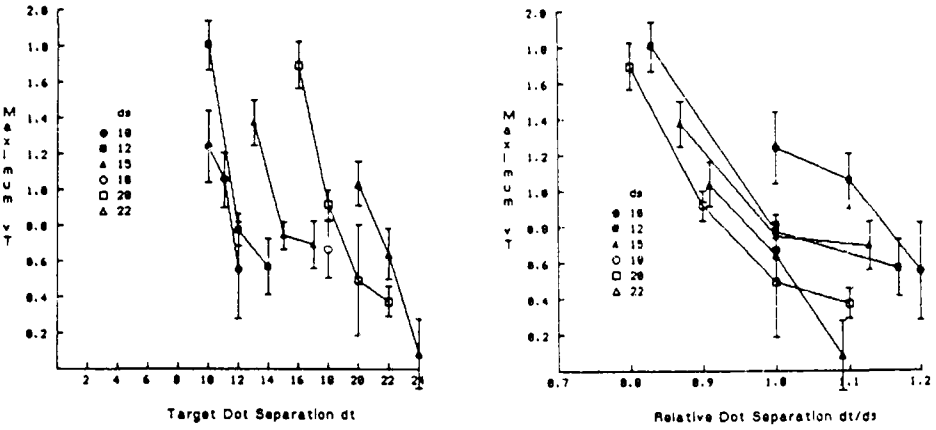


Figure 4. (a) Experimental maximum transverse variation v_T as a function of target dot separation d_t. Parameters are: $L = 200$, $v_L = 0$; d_s ranges from 10 to 22. Error bars represent one standard deviation. (b) The same curves plotted as a function of relative target dot separation d_t/d_s.

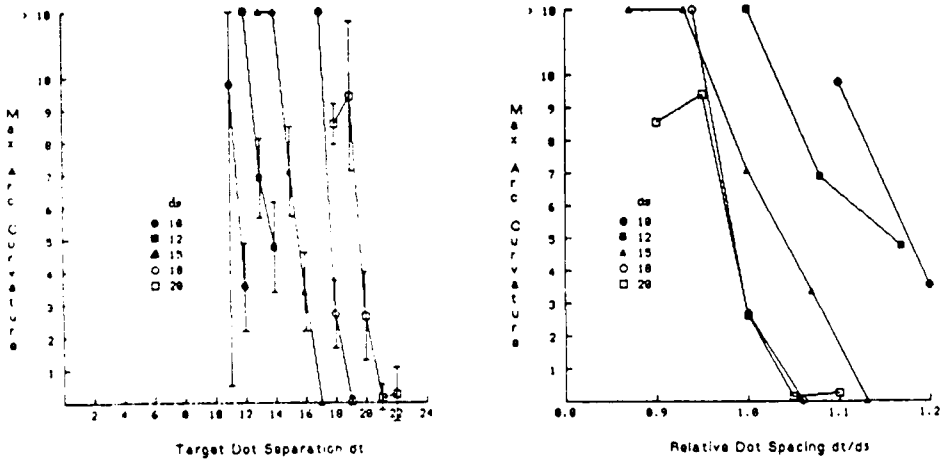


Figure 5. (a) Experimental maximum arc curvature κ as a function of target dot separation d_t. Plotted curvature is 1000 imes actual curvature, for ease of reading. Arc length $L = 200$. Error bars are one standard deviation. (b) The same curves as a function of relative dot separation d_t/d_s. Error bars are omitted for clarity.

model is based on comparing the dot density in a central target region with that in a surrounding region. The model uses operators that compute the probability of the number of dots in the center region, based on an estimate of the density of dots in the surrounding region. The smaller this probability, the less likely that event is to have occurred at random — i.e., the more likely it is to be non-accidental in origin.

The foundations of this approach have been laid by Lowe [8,10], Binford [3], and Witkin and Tenenbaum [18]. They claim that the goal of perceptual organization is to find non-accidental structures in images. Lowe suggests that perceptual organization be viewed as a process which assigns a degree of significance to various groupings of image features. Lowe's and Witkin and Tenenbaum's notion of non-accidentalness involves estimating the prior probability of occurrence of a pattern in the image, and measuring the accuracy with which a relation (e.g., collinearity) holds. The current approach is similar in spirit, but makes no use of prior probabilities as theirs does.

For the model to have relevance to real images, we must make a broadly applicable assumption about the distribution of dots in the image. One general assumption is that the dots are independently and uniformly spaced; this dot distribution is well modeled by a Poisson random distribution.

In detail, consider a rectangular *test region* in the image, of length L and width w, and a *surround region* surrounding it. Suppose there are n_c dots in the central test region. We form the null hypothesis H_0: the dots in the center region and in the surround come from the same distribution, and test it against the hypothesis H_1: the dots in the center region come from a distribution with greater mean. From the surround we estimate the surround dot density ρ. We can calculate the probability $P_{\text{random}}(n_c)$ of n_c dots by the Poisson probability

$$P_{\text{random}}(n_c) = \frac{\exp(-\rho wL)(\rho wL)^{n_c}}{n_c!}.$$

The expected number of dots in the region is ρwL; as n_c increases above this mean value, the probability that this event occurred at random decreases, and the probability it was non-accidental increases. We can plot the probability $P_{\text{random}}(n_c)$ as a function of n_c, for given ρ, w and L. If we keep L constant and increase w, the mean number of dots in the region increases, as does the variance of the distribution. Some members of this family of curves are shown in Fig. 6.

Suppose we choose a criterion of α (say, 0.001) for statistical significance. For any width w, we can find an n_w such that when $n_c \geq n_w$, $P_{\text{random}}(n_c) < \alpha$. As

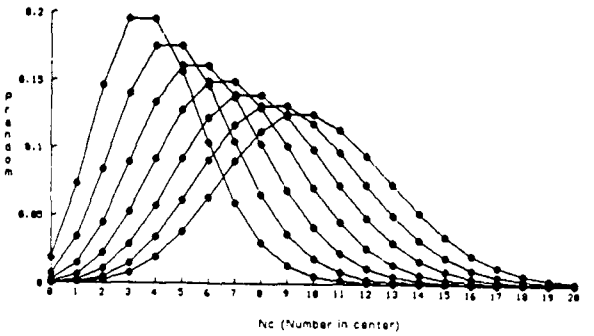


Figure 6. $P_{\text{random}}(n_c)$ as a function of n_c for several widths w. The widths range from 8 to 20; as the width increases, the curve shifts to the right and widens. Other parameters are: $\rho = 1/20^2 = .0025$ (corresponding to $d_s = 20$) and $L = 200$. Note that the required n_w for which $P_{\text{random}}(n_w) < \alpha$ increases as w increases.

w increases, so does the n_w for which statistical significance is reached: that is, more dots are required to maintain the same level of significance.

5 Predictions of the Model

Here I show how we can use the model to make predictions about line and curve detection.

5.1 Lines

Our goal is to find how the maximum v_T varies with target dot separation d_t. Suppose that some of the dots in the test region were generated by a process like the one used in the experiments described above: they are evenly spaced in a line of length L, separated by d_t and have a transverse variation v_T. The width of such a target is $d_t v_T$, and it contains $\lceil L/d_t \rceil$ dots. In order for all the target dots to fall in the test region, we require $w \geq d_t v_T$ or $v_T \leq w/d_t$; this provides the maximum v_T for these conditions.

Some dots from the surround will fall in the test region as well; the expected number of such dots is ρwL. If there are n_w dots in the test region, then the number of dots from the target is $m = n_w - \rho wL$. We then calculate $d_t = L/m$ and hence the maximum v_T is $w/d_t = wm/L$. Thus, if we know the number of dots n_w required to fall in the region for significance, we can find d_t and the maximum v_T for the corresponding dotted target line.

If we consider a number of widths w and fixed L, for a fixed surround density $\rho = 1/d_s^2$, we can find d_t and maximum v_T for each of these widths. Collecting these results, we obtain a curve of maximum v_T

versus d_t just as in the human experiments. We can compare these predictions with experimental results. I plot maximum variation v_T versus d_t in Fig. 7a; the several curves correspond to several different surround densities. Fig. 7b shows the same results as a function of relative dot spacing d_t/d_s. When plotted in this manner, the curves exhibit the same shape but do not coincide. The prediction is that lines in dense surrounds should be somewhat easier to detect than lines in sparse backgrounds. Comparing Fig. 7 with Fig. 4, we see that this prediction matches experimental results, but that this effect is less pronounced in human vision.

5.2 Curves

First, note that rectangular test regions can be used to detect dotted curves. When the arc curvature is low, a narrow region suffices to detect the arc; as the curvature increases, an increasingly wide region is necessary. As the region widens, more dots are required to fall in it in order to maintain the same level of significance, so the dots in the arc must be closer together. To calculate the maximum curvature for arcs, I use the same sort of analysis as applied above to line detection.

Consider a circular arc passing through a region of length L and width w, as depicted in Fig. 8. The arc of maximum curvature is obtained when the arc is situated as shown. The radius of the arc can be computed by analogy to (1): $R = [(L/2)^2 + w^2]/2w$, and the curvature is $\kappa = 1/R$. The angle θ is $\theta = \tan^{-1}[(L/2)/(R-w)]$.

As before, we determine the number of dots n_w needed to fall in the region in order to ensure a statistical significance criterion of α. For significance we need $m = n_w - \rho w L$ target dots, since an expected $\rho w L$ dots from the surround fall in the region as well. The dot separation d_t can then be computed as $d_t \approx 2R\theta/m$ (assuming $R \gg d_t$).

Thus, given a test region of size $L \times w$ and a surround density $\rho = 1/d_s^2$, we can calculate the dot spacing d_t and maximum curvature κ. Varying w yields a number of predicted data points; the results for several values of d_s are shown in Fig. 9a. As expected, the predicted maximum curvature increases as the dots in the arc fall closer together. Fig. 9b shows the predicted maximum curvature versus dot separation ratio d_t/d_s. Note that the curves are nearly coincident, but again there is a tendency for targets in dense patterns to be more easily detected than in sparse backgrounds. Human experiments show the same effect; compare Fig. 5.

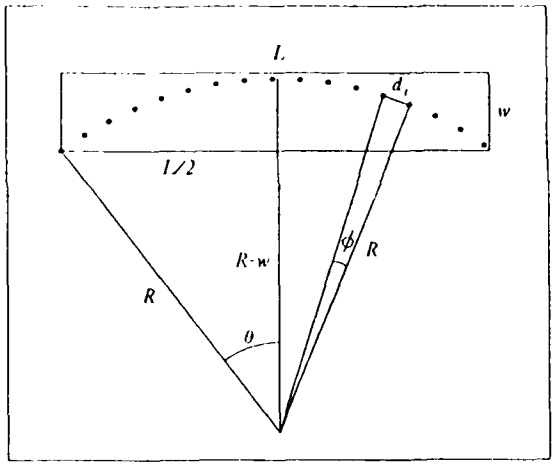


Figure 8. A dotted circular arc passing through a region of length L and width w. The arc radius is R and the angle between two dots in the arc is ϕ.

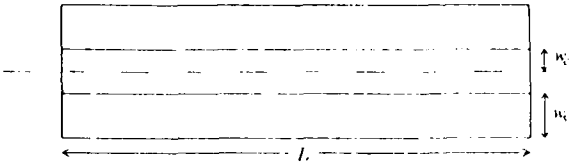


Figure 10. Geometry of an operator of the type used in the simulations.

6 Computer Simulations

To determine how well operators such as those proposed here would actually perform in a line-detection task, I performed computer simulations in which dotted line stimuli similar to those presented to human subjects were presented to a simulation system. The simulation system used operators of various sizes to try to detect the target. This section describes the simulations and their results. The details of the simulation setup appear in Appendix A.

The simulations are based on the assumption that the human visual system contains line-detection operators of various lengths and widths at each position in the visual field. As is commonly done in signal detection theory [6], I model the forced-choice detection task as follows. The program calculates the output of each operator at each position, and the response of the system for one trial corresponds to the position of that operator with the largest output. The program records detection accuracy; as v_T increases, performance deteriorates, as expected.

Each operator in the simulation (see Fig. 10) counts the number of dots in its center region and in its surround, n_c and n_s, and estimates the surround density

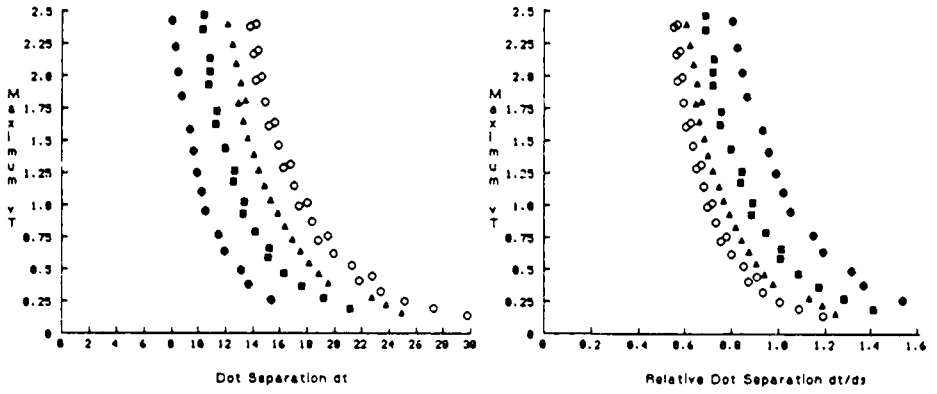


Figure 7. Predicted maximum transverse variation v_T as a function of target dot separation d_t. (a) The results for several surround densities, from left to right: $d_s = 10, 15, 20, 25$; $\rho = 1/d_s^2$. Probability criterion was $\alpha = 0.0001$, $L = 200$. (b) The same results, plotted as a function of d_t/d_s.

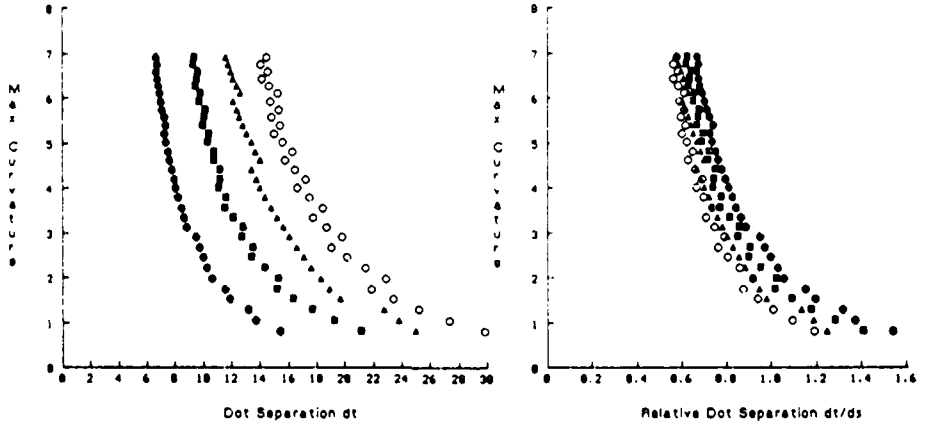


Figure 9. Predicted maximum curvature as a function of target dot separation. Plotted curvature is 1000 times actual curvature, for ease of reading. $L = 200$, $\alpha = 0.0001$. (a) Results for several surround densities are shown. From left to right, $d_s = 10, 15, 20, 25$. (b) The same results are plotted as a function of relative dot separation d_t/d_s.

$\hat{\rho} = n_s/(2w_sL)$. It then calculates the probability of those n_c center dots, assuming a Poisson distribution for the surround, using

$$P_{\text{random}}(n_c, n_s) = \frac{\exp(-2\hat{\rho}w_cL)(2\hat{\rho}w_cL)^{n_c}}{n_c!}.$$

Now

$$2\hat{\rho}w_cL = 2\left(\frac{n_s}{2w_sL}\right)w_cL = n_s\frac{w_c}{w_s},$$

so that

$$P_{\text{random}}(n_c, n_s) = \frac{\exp(-n_sr)(n_sr)^{n_c}}{n_c!} \qquad (3)$$

with width ratio $r = w_c/w_s$. The operators in the simulations compute the negative logarithm of (3). Since the simulations use only the operator with maximum response, and this is a monotonic transformation, it has no effect on the results.

I collected the simulation results for several values of d_s, as in the human experiments, and used the 80% threshold to find the maximum v_T for each value of d_s and d_t. These results are shown in Fig. 11. Note that the results are not scale invariant: targets in dense dot patterns (e.g., $d_s = 8$) are more easily detected than those in sparse patterns (e.g., $d_s = 16$). This effect seems somewhat more pronounced than in the human results, however; compare Fig. 4b.

6.1 Comparison with other operators

It is of interest to compare the results obtained with these significance-estimating operators with other kinds of operators. Difference-of-Gaussians (DOG) operators are commonly used in computer vision programs for edge detection, and Zucker [20,22] has used them for orientation selection. For computational efficiency, the difference-of-Gaussians is sometimes approximated by a difference-of-boxes (DOB) operator [3]. The output of these operators is their convolution with the image; they are linear filters that compute the difference of two convolutions. The output of the significance operator cannot be expressed as a convolution. The computational details of the DOG and DOB operators are relegated to Appendix B.

I ran some additional simulations with the difference-of-Gaussians and difference-of-boxes operators. Each of the three types of operator was evaluated on the same dot pattern; each operator type had the same variety of lengths and widths at four positions. Accuracy of detection was recorded as before. Some typical results are shown in Fig. 12; these are raw data showing accuracy versus v_T. Results for other stimulus parameters

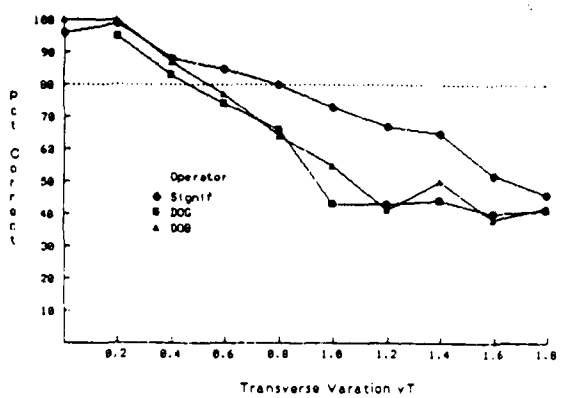


Figure 12. Results of the three kinds of operators on the same set of stimuli. Parameters for the stimuli were: $d_s = d_t = 12$, $L = 100$. Number of trials for each data point is 100. Curves for other parameter values show similar trends.

are similar. Results for the three operators are generally quite similar, but the significance operator performs somewhat better than the two convolution operators: the former is better able to detect linear patterns. Comparisons of maximum v_T thresholds show that all operators have the same qualitative characteristics (see Fig. 11). All the operators can detect lines in dense dot patterns more easily than in sparse patterns. The outputs of the DOG and DOB can be viewed as an approximation, perhaps, to a calculation based on the significance of the target line.

7 Discussion

The results presented here indicate that the statistical model is consistent with human performance on the preattentive tasks considered in my experiments. That does not imply, of course, that the human visual system uses this mechanism, or even if it does, that it uses no other mechanisms. I will be cautious and merely suggest that in this forced-choice task, under these impoverished conditions, a computation like the one suggested here may be in use.

The results also support the hypothesis that humans use elongated operators to detect curves, at least preattentively. However, more sophisticated experiments than those reported here would be needed in order to confirm or disprove that hypothesis. Other research [5,12,17] provides further evidence for this hypothesis.

The model presented here does not explain why a dotted line with many dots is no easier to detect than one with just five or six; Uttal [14] notes the same phenomenon. The model does explain why longitudinal variation v_L has little effect on detectability.

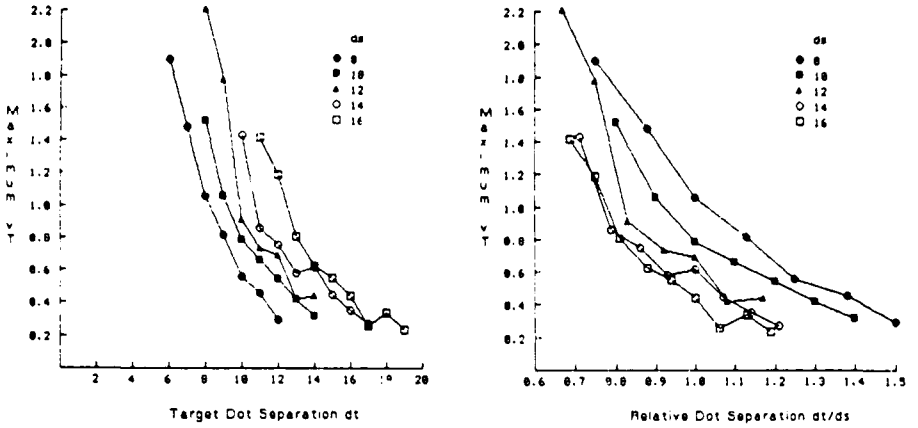


Figure 11. (a) Simulation results for linear targets. Curves correspond to constant values of d_s. (b) The same results plotted as a function of d_t/d_s.

Regardless of the psychological validity of this model, stronger claims can be made about its relevance to computer vision. Computing the probability of non-accidentalness of linear patterns is a powerful way of detecting them in an image. This method approaches the problem of how to determine when a pattern is a line and when it is not: it computes the statistical significance of a grouping relative to the local background. For straight lines with little noise in the surround, the significance of the grouping will be very high. For marginal lines such as the one depicted at the bottom of Fig. 1, the significance will be much lower. Depending on the goals of the vision system, it may be useful to look only for highly significant groupings, or it may be necessary to look for even marginally significant ones. The mechanism proposed here allows us to do either.

The reader should note that the simple predictions based on this statistical model (e.g., Figs. 7 and 9) fail to consider a number of factors. The predictions do not consider the number of alternatives for target location, nor do they include any notion of choosing between alternatives. A realistic system for line detection needs to measure the significance of line-like structures at several positions in the image, at several lengths and widths, and choose the position corresponding to the 'best' one as its choice in the forced-choice experiment. The predictions of this model indicate only the qualitative characteristics of a system using the significance-estimating approach.

Although the simulations do consider these factors, they fail to consider other potentially important ones. First, the human visual system does not know precisely the position of the target lines. The simulations assume that the positions of the targets are known with no uncertainty, and that other operators that may be nearby are ignored. Including the effects of such nearby operators may alter the results of the simulations. Second, the simulation system includes no model of how the outputs of the operators are used after the initial detection stage. Any vision system that uses such line detectors must include some later stage to further process their output. It is possible that adding this further stage of processing will result in a system that behaves more like the human visual system.

One of the contributions of this paper is to provide a theoretical foundation for finding structures by measuring their significance with respect to the surround. The model involves no thresholds, as some other models do. I have argued that using thresholds in this case is misleading and not useful; it is more valuable to measure the probability that the structure arose by accident, and reason further based on that information. Similar reasoning can be applied to detection of other features, like edges; it is the class of pattern being detected and the statistical significance of that pattern in the image that matters.

Many questions must be answered before this model can be embodied in an algorithm. It seems clear that there will be many detectors with various sizes, orientations, and widths operating at each image position; ideally, they would operate in parallel for speed. The parameters of the detectors need to be chosen. Also, the outputs of all the operators must be combined in some coherent fashion. This is simplified since we know exactly what the operators are computing from the image. In contrast, some existing algorithms take an ad hoc approach, with arbitrarily selected operators and arbitrary parameters, and where the output has no clear meaning; these linking algorithms and the operators they use often have little theoretical justification.

The approach suggested here promises to clear up some of that confusion.

8 Related Work

Here I briefly summarize research related to the current problem. This work is motivated to a large extent by Lowe's work [8,9,10] on finding groupings in images. His demonstration [8] that widely-separated dots cannot easily be seen when surrounded by noise dots provides a starting point for the psychophysical experiments reported here. Lowe's work [8,9] on finding curvilinear structure in images was designed primarily to find structures in dots that have already been linked on the basis of proximity, rather than on sparse dotted lines surrounded by noise.

Lowe [9] claims that one of the goals of perceptual organization is to statistically distinguish between accidental and non-accidental instances of a relation. Witkin and Tenenbaum [18] claim that the non-accidentalness principle unifies several problems in perceptual organization, and is in fact the general goal of image organization. One fruitful assumption is that the purpose of perceptual organization is to detect stable image groupings that arise due to structure of the scene rather than accident. In the context of grouping dots into lines and curves, discriminating between accident and non-accident means determining the likelihood that the structures arose due to random placement of dots. Structures that are unlikely to be accidental have a causal origin, i.e., they are the projection of a corresponding structure in the scene. Lowe notes that "perceptual organization can be viewed as a process which assigns a degree of significance to each potential grouping of image features" [8]. This suggestion is taken rather literally in this paper, where each operator computes the statistical significance of the structure within its receptive field. These points are elaborated by Lowe [8] and Binford [4,3].

Zucker and his colleagues [19,20,21,22] distinguish between Type I and Type II processes in dot grouping: Type I processes infer 1D contours, while Type II processes deal with 2D flow fields. Although Zucker apparently does not consider the influence of background noise on the inference of Type I contours, the problem considered in this paper falls on the Type I side of his dichotomy.

Zucker's papers present a model for finding oriented entities corresponding to Type I or Type II patterns. The model is suggested by biology: first, convolution of the image with orientationally selective receptive fields; and second, interpretation of the outputs of those operators. The operators in his implementation [22] compute the difference of Gaussians; they have several arbitrary parameters, and their output lacks a clear semantic meaning. In contrast, the output of the operators proposed here has a clear interpretation. It is difficult to determine how Zucker's methods would perform on the kinds of patterns considered here. Zucker [20] proposes that a vector field of orientations be found in the image by applying these operators and interpreting their outputs; from this vector field, he proposes that Type I contours be found by solving a set of differential equations. This procedure assumes that there is only one set of curves passing through the image, that there is no ambiguity. It is often the case, however, that there is no perceptually unambiguous way to link a set of dots (or edgels); there may be many ways to impose groupings on the input. (How many ways are there to group the dots in Figs. 1 or 2 into curves?) A system that measures the significance of different groupings allows this flexibility, while Zucker's method does not seem to.

9 Summary and Conclusions

I presented the results of psychophysical experiments that show the effect of point scatter and curvature on our ability to detect dotted lines and curves. The main factor that determines our ability to detect such patterns is the relative separation of target dots and surround dots, but there is a significant effect of density: we can detect curves with more jaggedness and curvature in dense backgrounds than in sparse.

I presented a statistical model for line and curve detection, and used the model to make qualitative predictions about human performance in this task. The model is based on estimating the density of dots surrounding an elongated center region, counting the number of dots in the center, and calculating the probability that those dots fell in the center region at random. When this probability is low, we infer that the dots in the center region have a non-accidental origin, i.e., correspond to some structure in the scene. The predictions based on this model agree, within their limits, with the empirical results. Predictions for curve detection, based on detection of lines, agree with data on human curve detection, supporting a hypothesis that curves are detected by line detectors.

I discussed computer simulations, in which line-detection operators of the type proposed here are applied to a forced-choice line detection task. The results were discussed and compared to human results.

The model discussed here promises to lay a theoretical foundation for the detection of lines and curves in images. In a future paper, I will discuss the imple-

mentation of an algorithm based on this model, and its application to the task of finding curvilinear structures in images.

10 Acknowledgements

My advisor T. Binford provided many insights into this problem, helped with the analysis, and carefully read a draft. B. Wandell was a great help throughout, especially with the psychometric analysis, and supplied the program that fit psychometric functions to the data. A. Witkin helped crystallize some of the ideas. D. Lowe provided comments on a draft. E. Isaacs was a willing experimental subject. This research was supported by ARPA contract N00039-84-C-0211.

APPENDIX A. Details of Simulation Setup

The simulation setup consists of a number of differently-shaped rectangular operators at each of four positions and orientations corresponding to the possible target positions. The scale of this simulation is approximately half that of the human experiments: the image is 200 pixels square, and the target length L is 100. At each position, the operators come in a variety of lengths and widths. There are four operator lengths, ranging evenly from 50 to 170. For each length, there are four widths specified as a fraction of the length; these fractions range evenly from 0.01 to 0.08. Thus there are 16 operators at each position, 64 altogether.

The surround width of each operator is fixed at three times the center width. This is a compromise between efficiency and accuracy; a larger w_s slows down the simulations. Also, a larger surround increases the chance of including some qualitatively different area of the background. There is a tradeoff between random error due to small sample size (small w_s) and systematic error due to sampling the wrong region [3]. A typical operator of this sort appears in Fig. 10.

It is possible that a different coarseness of length and width resolution in the assortment of operators might affect the simulation results. However, pilot experiments showed that four or five different lengths and widths are sufficient for full performance; using more than this does not increase performance but merely slows down the simulations.

APPENDIX B. Construction of DOG and DOB operators

The difference-of-Gaussians operator convolves the function $G(x) = g_c(x) - g_s(x)$ with the dot image; each dot has the value 1. The functions $g_i(x)$ are defined by

$$g_i(x) = w_i \exp(-k_i^2 x^2)$$

where

$$w_i = 1/\sqrt{2\pi}\sigma_i \quad \text{and} \quad k_i^2 = 1/2\sigma_i^2.$$

The difference-of-boxes operator computes the convolution $H(x) = h_c(x) - h_s(x)$, where $h_c(x)$ and $h_s(x)$ are defined by

$$h_c(x) = \begin{cases} 0, & \text{if } |x| > w_c; \\ \alpha, & \text{if } |x| \le w_c. \end{cases}$$

$$h_s(x) = \begin{cases} 0, & \text{if } |x| > w_c + w_s; \\ \beta, & \text{if } |x| \le w_c + w_s. \end{cases}$$

For zero mean, we require $\alpha w_c = \beta(w_c + w_s)$.

For a reasonable comparison of operator performance, it is necessary to match the center and surround regions of each operator as closely as possible. For the difference-of-boxes operator and the Poisson significance-estimating operator, this is easy since the regions are strictly delimited. For the difference-of-Gaussians operator, however, the surround region is theoretically the entire image; however, dots far from the center are given very low weight in calculating the response. The DOB and DOG operators are differences of weighted sums calculated over center and surround convolution regions. These regions can be matched well in size by making the standard deviations of their defining functions equal.

The variance of a function $f(x)$ on an interval $[a, b]$ is defined as the mean squared error $\overline{(x-\mu)^2}$, where μ is the average value of x in $[a, b]$. To find the variance of the function $f(x) = \alpha$ on $[-w, w]$, we note that $\mu = 0$ since the interval is centered on zero, so the variance is $\overline{x^2}$. The average value $\overline{g(x)}$ of a function $g(x)$ on $[a, b]$ is defined as

$$\frac{1}{b-a}\int_a^b g(x)dx$$

so the variance of f is

$$\frac{1}{2w}\int_{-w}^{w} x^2 dx = \frac{1}{2w}\frac{x^3}{3}\Big|_{-w}^{w} = \frac{w^2}{3}$$

and the standard deviation is just $w/\sqrt{3}$. Thus, in these comparison simulations, σ_c for the DOG was set to $w_c/\sqrt{3}$ and σ_s was set to $(w_c + w_s)/\sqrt{3} = 4w_c/\sqrt{3}$

(since the surround width was three times the center width in the simulations).

The difference-of-boxes operator and the difference-of-Gaussians operator perform similarly since they have nearly the same shape. That is, since they are constructed to have the same mean and same variance (second moment) and are even functions, their first and third moments are zero. Thus, the first moment in which they differ is the low order fourth moment, and they perform nearly equivalently.

References

[1] D.H. Ballard and C.M. Brown. *Computer Vision.* Prentice-Hall, Englewood Cliffs, NJ, 1982.

[2] I. Biederman. Human image understanding: Recent research and a theory. *Comp. Vision, Graphics, and Image Proc.*, 32(1):29–73, 1985.

[3] T.O. Binford. Figure/ground: Segmentation and aggregation. In O.J. Braddick and A.C. Sleigh, editors, *Physical and Biological Processing of Images*, Springer-Verlag, New York, 1983.

[4] T.O. Binford. Inferring surfaces from images. *Artificial Intelligence*, 17:205–244, 1981.

[5] C. Blakemore and R. Over. Curvature detectors in human vision? *Perception*, 3:3–7, 1974.

[6] C. Coombs, R. Dawes, and A. Tversky. *Mathematical Psychology: An elementary introduction.* Prentice-Hall, Englewood Cliffs, NJ, 1970.

[7] B. Julesz. *Foundations of Cyclopean Perception.* The University of Chicago Press, London, 1971.

[8] D. Lowe. *Perceptual Organization and Visual Recognition.* Kluwer Academic Publishers, Boston, 1985.

[9] D. Lowe. *Three-dimensional object recognition from single two-dimensional images.* Technical Report 202, Courant Institute, New York University, 1986.

[10] D. Lowe. Visual recognition from spatial correspondence and perceptual organization. In *Proc. Int. Joint Conf. on Art. Intell.*, pages 953–959, 1985.

[11] V.S. Nalwa and E. Pauchon. Edgel-aggregation and edge-description. In *Image Understanding Workshop*, 1985.

[12] B.N. Timney and C. Macdonald. Are curves detected by 'curvature detectors'? *Perception*, 7:51–64, 1978.

[13] A. Treisman. Preattentive processing in vision. *Comp. Vision, Graphics, and Image Proc.*, 2(31):156–177, 1985.

[14] W.R. Uttal, L.M. Bunnell, and S. Corwin. On the detectability of straight lines in visual noise. *Perception and Psychophysics*, 8(6):385–388, 1970.

[15] R. Vistnes. Detecting structure in random-dot patterns. In *Image Understanding Workshop*, 1985.

[16] A. Watson. Probability summation over time. *Vision Res.*, 19:515–522, 1979.

[17] H.R. Wilson. Discrimination of contour curvature: Data and theory. *J. Opt. Soc. Am.*, A 2(7):1191–1198, 1985.

[18] A.P. Witkin and J.M. Tenenbaum. On the role of structure in vision. In O.J. Braddick and A.C. Sleigh, editors, *Physical and Biological Processing of Images*, Springer-Verlag, New York, 1983.

[19] S.W. Zucker. Computational and psychophysical experiments in grouping: Early orientation selection. In J. Beck, B. Hope, and A. Rosenfeld, editors, *Human and Machine Vision*, Academic Press, New York, 1982.

[20] S.W. Zucker. Cooperative grouping and early orientation selection. In O.J. Braddick and A.C. Sleigh, editors, *Physical and Biological Processing of Images*, Springer-Verlag, New York, 1983.

[21] S.W. Zucker. *The diversity of perceptual grouping.* Technical Report TR-85-1R, Comp. Vision and Robotics Lab, McGill University, April 1985.

[22] S.W. Zucker. Early orientation selection: Tangent fields and the dimensionality of their support. *Comp. Vision, Graphics, and Image Proc.*, 32(1):74–103, 1985.

[23] S.W. Zucker, K.A. Stevens, and P. Sander. The relation between proximity and brightness similarity in dot patterns. *Perception and Psychophysics*, 34(6):513–522, 1983.

LEARNING SHAPE COMPUTATIONS

John (Yiannis) Aloimonos
David Shulman

Center for Automation Research
University of Maryland
College Park, MD 20742

ABSTRACT

Descriptions of physical properties of surfaces that we see, such as their shape, distance, boundaries and discontinuities, must be recovered from the image data. A large part of research in computational vision aims to understand how such descriptions can be obtained from noisy and ambiguous data. We view these problems as ill-posed problems and we present a unified theory for the computation of shape from several cues, such as shading, texture, contour or motion. We study how a connectionist network (neural network) can learn all the parameters that are involved in the computation of shape from any cue, and then how it can, through an iterative technique that converges to a unique value, compute the solution of any "shape from X" problem.

1. INTRODUCTION

One of the main goals of computational vision is to develop image understanding systems that automatically construct scene descriptions from the image data (input). Early vision consists of a set of processes that recover physical properties of the visible three dimensional surfaces from the two dimensional intensity arrays. Such properties include shape, distance, reflectance and others. On the other hand many cues are available and they are used by different modules in the recovery process. Such cues are texture, contours, shading, color, stereo and motion. Much current research has analyzed processes of early vision, such shape from shading [16], shape from texture [17, 18, 19], shape from motion (kinetic depth effect) [20] and shape from contour [21, 22]. Several successful algorithms have been proposed for the above problems that worked well for the assumptions that they were based on. In general, all these different "shape from X" problems have been studied separately, and only with respect to one or two levels of a visual system, as defined by Marr [23].

We show here that all the problems of computing shape from shading, texture, contour and motion may be solved by the same neural network that iteratively converges to a unique solution. The network solves the different problems merely by changing the weights in the connections of the different units. In the course of our analysis we regularize some ill-posed problems that are nonlinear and for which standard regularization techniques [24] are not applicable.

2. MOTIVATION

All the problems of shape from shading, contour, texture and motion have the characteristic that they are ill-posed in the sense of Hadamard. A problem is well-posed when its solution exists, it is unique and depends continuously on the initial data. Ill-posed problems fail to satisfy one or more of these criteria. The main idea of "solving" ill-posed problems, i.e. to restore "well-posedness", is to introduce suitable *a priori* knowledge that will restrict the space of admissible solutions. *A priori* knowledge can be exploited, for example, under the form of constraints on the possible solutions. The regularization of the ill-posed problem of finding w (the world) from the data i (image) and the constraint $Aw = i$ when A is not invertible (ill-posed problem), amounts to choosing a norm $\|\cdot\|$ and a functional Pw, and then applying an optimization technique. For example, we can find the w that minimizes the function $\|Aw - i\|^2 + \lambda\|Pw\|^2$, where λ is the so called regularization parameter. This approach was initiated in computer vision by Poggio and his colleagues [25]. A should be a linear operator, the norms quadratic and P linear. When the constraint is nonlinear, and it is nonlinear most of the time, then finding a unique solution is not guaranteed [28]. Finally, our learning scheme was motivated by Poggio's work [37], where it was observed that regularizing operators might be learned without the need for explicit variational (smoothness) conditions.

The organization of the paper is as follows: The next section develops the computational theory of shape computation from contour, texture, motion and shading, i.e. what are the constraints that govern the particular problem. In other words we develop the first level of a visual system (as proposed by Marr). Section 4 develops an algorithm for shape computation and studies its properties, i.e. the second level of a visual system. Finally, Section 5 discusses implementation of shape computation algorithms (the third level) and develops a theory for learn-

ing how to compute shape from different kinds of cues in a neural net.

3. COMPUTATIONAL THEORY OF SHAPE

Here we study the physical constraints that are involved in the computation of shape from different cues. Each case is studied separately.

3.1. Computational theory of shape from shading

This theory has been developed by Horn and his colleagues [16]. The treatment is done under orthography (actually, in the perspective case the same laws apply, as noted in [26]). According to this theory, the intensity at a point (x,y) of an image is given by the formula

$$I(x,y) = \rho \vec{s} \cdot \vec{n}$$

where ρ is a constant depending on the material of the surface in view, $\vec{s}$ is the direction of the point light source and $\vec{n}$ the surface normal of the surface point whose image is the point (x,y). If we are trying to compute shape, then we really are after two parameters (p,q) at every image point (x,y), where p,q is the gradient of the surface normal vector at the surface point whose image is (x,y). The above equation, when expressed in terms of the gradient (p,q), becomes

$$I(x,y) = \frac{\rho(pp_s + qq_s + 1)}{\sqrt{(1 + p^2 + q^2)(1 + p_s{}^2 + q_s{}^2)}}$$

where $\vec{s} = (p_s, q_s, -1)$. It is obvious that at every point of our image we have two unknowns and only one constraint.

3.2. Computational Theory of Shape from Motion

Here we investigate the constraints imposed by motion. This problem has been extensively studied in the field of structure from motion or kinetic depth effect, for both orthography and perspective projection and for discrete or continuous motion. Here we use orthography and discrete motion. Also, we consider the object in view as consisting of features, i.e. points of interest; so we speak of the object as a cloud of points that moves. Furthermore, only rigid motion is considered. In the pionnering work of Ullman it was proved that three orthographic projections of four noncoplanar points admit one solution for the structure and motion of the points. It was stated in [23] that two orthographic projections of any number of points admit infinitely many interpretations for the structure of the points. This was formally proved in [27], correcting a previous result. Here, we study the constraint between displacements of points and local shape. It turns out that this constraint is of the same form as in the shape from shading case, i.e. a conic in the gradient (p,q). For this we need the following lemma:

Lemma :

Given two distinct orthographic projections of three points in a rigid configuration, the gradient (p,q) of the plane that the three points define (with respect to the coordinate system of the first frame), lies on a conic section in gradient space. The coefficients of this conic section depend entirely on the interframe displacements of the points.

Proof:

Let the three points in space be O,A,B in their first positions and O',A',B' in their second positions and their projections in the two frames be O_1,A_1,B_1 and O_2,A_2,B_2, respectively. Let also the gradient of the plane OAB be $G = (p,q)$. Furthermore, let

$$\vec{O_1A_1} = \vec{\alpha_1} = (x_1, y_1) \tag{1}$$

$$\vec{O_1B_1} = \vec{\beta_1} = (c_1, d_1) \tag{2}$$

$$\vec{O_2A_2} = \vec{\alpha_2} = (xsub2, y_2) \tag{3}$$

$$\vec{O_2B_2} = \vec{\beta_2} = (csub2, d_2) \tag{4}$$

Considering the geometry of the first projection (OAB to $O_1A_1B_1$), we have

$$\vec{OA} = (x_1, y_1, \vec{G}.\vec{\alpha_1}) \tag{5}$$

$$\vec{OB} = (c_1, d_1, \vec{G}.\vec{\beta_1}) \tag{6}$$

Similarly, considering the second projection (OAB to $O_2A_2B_2$), we get

$$\vec{O'A'} = (x_2, y_2, \lambda) \tag{7}$$

$$\vec{O'B'} = (c_2, d_2, \mu) \tag{8}$$

where λ and μ are to be determined.

But, because of the rigid motion, the vectors $\vec{OA}$ and $\vec{O'A'}$ have the same length. The same holds for the vectors $\vec{OB}$ and $\vec{O'B'}$. From these requirements we get

$$\lambda = \pm\sqrt{\vec{\alpha_1}^2 + (\vec{G}.\vec{\alpha_1})^2 - \vec{\alpha_2}^2} \tag{9}$$

$$\mu = \pm\sqrt{\vec{\beta_1}^2 + (\vec{G}.\vec{\beta_1})^2 - \vec{\beta_2}^2} \tag{10}$$

Finally, again because of the rigidity, the angles between the vectors $\vec{OA}, \vec{OB}$ and $\vec{O'A'}, \vec{O'B'}$ are the same. From this, we get:

$$\vec{OA}.\vec{OB} = \vec{O'A'}.\vec{O'B'} \tag{11}$$

where . denotes the dot product operation.

Substituting in equation (11) from equations (5), (6), (7), (8), (9), (10), we get

$$\vec{\alpha_1}.\vec{\beta_1} + (\vec{G}.\vec{\alpha_1})(\vec{G}.\vec{\beta_1}) = \vec{\alpha_2}.\vec{\beta_2} \pm \lambda\mu$$

and substituting the values for λ and μ and squaring appropriately, we get

$$\begin{aligned}(\vec{\beta_1}^2 - \vec{\beta_2}^2)(\vec{G}.\vec{\alpha_1})^2 + (\vec{\alpha_1}^2 - \vec{\alpha_2}^2)(\vec{G}.\vec{\beta_1})^2 - 2(\vec{\alpha_1}.\vec{\beta_1} - \vec{\alpha_2}.\vec{\beta_2})(\vec{G}.\vec{\alpha_1})(\vec{G}.\vec{\beta_1}) \\ + (\vec{\alpha_1}^2 - \vec{\alpha_2}^2)(\vec{\beta_1}^2 - \vec{\beta_2}^2) - (\vec{\alpha_1}.\vec{\beta_1} - \vec{\alpha_2}.\vec{\beta_2})^2 = 0\end{aligned} \tag{12}$$

Given that

$$\vec{G}.\vec{\alpha_1} = px_1 + qy_1$$

and

$$\vec{G}.\vec{\beta}_1 = pc_1 + qd_1$$

the above equation (12) is of the form

$$Ap^2 + Bq^2 + \Gamma pq + \Delta = 0$$

where the coefficients A,B,Γ,Δ depend on the image vectors $\vec{\alpha}_1$, $\vec{\alpha}_2$, $\vec{\beta}_1$ and $\vec{\beta}_2$ (q.e.d.).

We can now easily establish the relationship between retinal displacements and local shape, by applying the previous lemma for the points (x,y), $(x + dx,y)$, $(x,y + dy)$. Then, assuming that the surface is locally planar and that the plane is defined by the three points in the world whose images are the points (x,y),$(x + dx,y)$,$(x,y + dy)$, we can derive a constraint between local shape and retinal motion.

Consider a moving surface $Z = Z(x,y)$ and let $[\Delta u(x,y), \Delta v(x,y)]$ be the discrete displacement field for two time instants t_1 and t_2 with $t_1 < t_2$, i.e. if an image point is at the position (x,y) at time t_1, then at time t_2 it will be at the position $[x + \Delta u(x,y), y + \Delta v(x,y)]$. Then, assuming that the surface is locally (differentially) planar, i.e. the points (x,y), $(x + dx, y)$, $(x,y + dy)$ define a plane, we can prove that the gradient (p,q) at a surface point whose projection on the image plane is the point (x,y), satisfies the following conic constraint:

$$k_1p^2 + k_2q^2 - 2k_3\,pq + k_4 = 0$$

with

$$k_1 = (\vec{\alpha}_1^2 - \vec{\alpha}_2^2)dx^2$$

$$k_2 = (\vec{\beta}_1^2 - \vec{\beta}_2^2)dy^2$$

$$k_3 = (\vec{\alpha}_1\vec{\beta}_1 - \vec{\alpha}_2\vec{\beta}_2)dx\ dy$$

$$k_4 = (\vec{\alpha}_1^2 - \vec{\alpha}_2^2)(\vec{\beta}_1^2 - \vec{\beta}_2^2) - (\vec{\alpha}_1\vec{\beta}_1 - \vec{\alpha}_2\vec{\beta}_2)^2,$$

where $\vec{\alpha}_1 = (0, dy)$, $\vec{\beta}_1 = (dx, 0)$

$$\vec{\alpha}_2 = [\Delta u(x,y + dy) - \Delta u(x,y),\ dy + \Delta v(x,y) + dy) - \Delta v(x,y)]$$

and $\vec{\beta}_2 = [dx + \Delta u(x + dx, y) - \Delta u(x,y),\ \Delta v(x + dx, y) - \Delta v(x,y)]$

The proof of the above equation is immediate from the application of the lemma to the points (x,y), $(x + dx, y)$ and $(x,y + dy)$. It has to be emphasized that the coefficients k_i, $i = 1, \ldots, 4$ are not constant; they are functions of (x,y) and so we write $k_i(x,y)$ instead of k_i.

3.3. Computational Theory of Shape from Patterns

Imagine that we are viewing a surface covered with repeated elements (texels or patterns). Then, the problem of shape from patterns is solving for the shape of the surface in view from the apparent distortion of the patterns. Several researchers have worked on this problem. Research reported in [31, 32, 33] made the assumption that the patterns were all identical and symmetric, and then the problem could be approached through skewed symmetry constraints. Research in [34] dealt with the case of planar surfaces and did not assume any pattern symmetry. Finally, research in [35] attempted the solution of this problem under the assumption that all patterns should be of the same area (which is true when all the patterns are the same). Symmetry was not a requirement. Furthermore, every texel is assumed to lie on a planar surface, and this means that the size of the texels on the surface has to be small compared with the change of surface orientation. Also, in this work the scene was assumed to be projected onto the retina under an approximation of perspective projection, called paraperspective. Paraperspective projection is analyzed in [27] and its approximation error is calculated. For the purposes of this section, we will only state the constraint relating the area of a world texel to the area of its image. Let S_I be the area of a texel in the image plane. Let S_W be the area of the texel on the world surface. Let p,q the gradient of the plane on which the world texel lies, and (A,B) the center of mass of the image texel. Then the following relation holds

$$S_I = \frac{S_W}{\beta^2}\frac{1 - Ap - Bq}{\sqrt{1 + p^2 + q^2}}$$

The quantity $\frac{S_W}{\beta^2}$ is called textural albedo, and in [27] methods are presented for its computation. For our purposes we will assume that the textural albedo is known and we will symbolize it by λ. So, in the case of shape from pattern, the constraint that governs the problem is

$$S_I = \lambda\frac{1 - Ap - Bq}{\sqrt{1 + p^2 + q^2}}$$

It is important to note that the constraint in this case is again a conic section in gradient space.

3.4. Computational Theory of Shape from Texture

This case is very similar to the previous one. Here we only consider a dot texture, i.e. we assume that the surface in view is co ered by dots. Shape cannot be recovered in this case from a monocular static view of the surface, unless some assumptions about the texture are made. The assumption of uniform density (or slight variations of it) has been shown to be successful in past research, for the case of planar surfaces. Here we assume a nonplanar surface, and we make the assumption that a unit area on the surface in view contains the same number of texels (points). In this case, the constraint relating local shape and density of the texture can easily be established and it is of a form similar to the previous case, but instead of areas, we have texture densities. We won't analyze this case in detail, since it follows easily from the previous section.

3.5. Summary

Up to this point we have analyzed the computational theory behind the processes of shape from shading,

motion, patterns and texture. In all cases, it was shown that the constraint is of the form

$$A(x,y)p^2 + B(x,y)q^2 + C(x,y)pq + D(x,y)p + E(x,y)q + F(x,y) = 0 \quad (13)$$

where A,B,C,D,E,F are functions of the position in the image and depend on the particular physical parameters (data). It has to be noted that the approximations made (paraperspective projection, orthographic projection) are not essential for our theory. If perspective projection were used then a more complicated constraint (still nonlinear of higher degree) would appear. But as will become evident later, the form of the constraint is not very essential, as long as it satisfies some weak constraint that will be described later.

3.6. Shape Representation: Stereographic Projection

Surface orientation is quantified by the surface normal, a unit vector in $\mathbf{R}^3$. Let (l,m,n) be a vector denoting the direction of a surface normal. Then this direction is defined by $(p,q) = \left(-\frac{l}{n}, -\frac{m}{n}\right)$. This quantity is called the gradient of the normal and the space of all the possible gradients defines the so-called gradient space. Gradient space is unbounded. But the gradient is not the only representation for surface orientation. Another representation that results in a bounded space is stereographic coordinates. A surface normal can be represented by a point on a unit sphere, called the *Gaussian sphere*. The part of the surface facing us corresponds to one hemisphere, let us define it to be the northern hemisphere, while points on the occluding boundaries correspond to the points on the equator. A point n on the Gaussian sphere corresponds to a unique gradient (p,q) and vice versa. This can be seen geometrically in the following way. Consider the projection of the Gaussian sphere by rays from its center onto a plane tangent to the north pole. This plane represents the gradient space (p,q). This projection is called the gnomonic projection and it has the property that great circles are mapped into lines. The north pole is mapped onto the origin of the gradient space, the northern hemisphere is mapped into the whole $p - q$ plane and points on the equator end up at infinity. Points on the lower hemisphere are not projected onto the $p - q$ plane. With the gradient space formalism, the equator maps into infinity. As a result, occluding boundary information cannot be expressed in gradient space. One solution to this problem is the use of stereographic projection. We can think of this projection in geometric terms also as a projection of the Gaussian sphere onto a plane tangent to the north pole. However, this time the center of projection is the south pole, not the center of the sphere. We label the axes of the stereographic plane as f,g. It can be shown that the relation between the stereographic space coordinates f,g and the gradient space coordinates is given by

$$f = 2p\left[\sqrt{1 + p^2 + q^2} - 1\right]/(p^2 + q^2)$$

$$g = 2q\left[\sqrt{1 + p^2 + q^2} - 1\right]/(p^2 + q^2)$$

$$p = \frac{4f}{4 - f^2 - g^2}$$

$$q = \frac{4g}{4 - f^2 - g^2}.$$

This mapping is conformal and the whole northern hemisphere is mapped onto a closed disc of radius 2 on the $f - g$ plane. The orientations of occluding boundaries correspond to points on the circumference of that disc. The advantage of the stereographic projection is that it maps the whole northern hemisphere (visible orientations) onto a bounded space, i.e. a disc of radius 2 in the stereographic plan with center at the origin (the point where the stereographic plane is tangent to the sphere, i.e. the north pole), and occluding boundary information can be easily expressed (the points of the circumference of the disc).

4. ALGORITHMIC THEORY OF SHAPE COMPUTATION

It is by now clear that in all the above cases (shading, texture, motion and pattern) the problem cannot be solved uniquely, unless an additional assumption is imposed. In this section we show that if we regularize these problems, i.e. if we impose an additional constraint (which is physically plausible), then we can uniquely solve these problems. If we use the equations that give the stereographic coordinates in terms of the gradient, we can substitute p,q in terms of f and g and assuming that the surface does not contain points whose orientation is equal to the orientation at the boundaries, we get a relation of the form

$$L(f,g,x,y) = 0. \quad (14)$$

with L a polynomial in f,g whose coefficients depend on the image data. It has to be noted that in all the problems of shape from shading, texture, patterns or motion, the constraint will be of the above form.

Clearly, all the above-mentioned problems of finding structure f,g from equation (14) are ill-posed problems in the sense of Hadamard [36], since the solution is not unique. To regularize the problem and make it well-posed, we should introduce some more constraints on the problem. Specific regularization methods for solving ill-posed problems have been developed [16, 24]. Following the paradigm of regularization theory, we require that the surface in view be smooth and we wish to minimize the quantity [37]

$$e = \iint_D \left\{ \left[(f_x)^2 + (f_y)^2 + (g_x)^2 + (g_y)^2\right] + \lambda(L)^2 \right\} dx\, dy \quad (15)$$

where λ is a regularization constant.

In other words, to solve this ill-posed problem (i.e. to make it well-posed), we should restrict the class of admissible solutions by introducing suitable *a priori* knowledge. This *a priori* knowledge can be exploited, for example,

under the form of variational principles that impose constraints on the possible solutions. By choosing to minimize e in (15) we wish to find a solution that best satisfies the constraint $L = 0$ and that is smooth, where the relative degree of smoothness is measured by the first term in (15). But this problem, even though it fits in the regularization paradigm, cannot be trivially solved, because of the nonlinearity involved. Here we present an algorithm that obtains a unique solution.

We require that the solution be a surface that best satisfies the motion constraint (eq. 14) and at the same time is as smooth as possible, according to the stabilizing functional [24] introduced in (15). This fits exactly into the regularization paradigm in vision, which originated in [25] and has received attention lately.

The question that arises is whether or not there exists a unique surface that minimizes e, and if so, to give an algorithm that finds this unique solution. In the sequel we investigate this question. Our analysis is done for the discrete case, i.e. by taking into account the discrete nature of images.

4.1. Finding the Unique Solution

Here we use help from widely known techniques in the area of partial differential equations, which have been applied to some computer vision research [29]. Consider the constraint equation $L(f,g,x,y) = 0$, $(x,y) \in D$, where D is the unit square region in the $x - y$ plane with mesh size m, and discretize e by using difference operators instead of differential operators and summations instead of integrals. We assume that the boundary of the object (image) is a square. This is done for simplicity here and without loss of generality. Generalization to any kind of boundary is possible, but tedious and will be reported elsewhere. Let $n = k^2$, where $k + 1 = \frac{1}{m}$. The desired surface is the one that minimizes

$$e = \sum_{i,j}(s_{i,j} + \lambda l_{i,j}) \tag{16}$$

where

$$s_{i,j} = \frac{1}{m^2}\Big\{[f_{i+1,j} - f_{i,j}]^2 + [f_{i,j+1} - f_{i,j}]^2 + \\ + [g_{i+1,j} - g_{i,j}]^2 + [g_{i,j+1} - g_{i,j}]^2\Big\}$$

$$l_{i,j} = \Big[L(f_{i,j}, g_{i,j}, i,j)\Big]^2$$

and where $f_{i,j}$, $g_{i,j}$ represent the surface orientation at the regular grid point (im,jm). This minimization is subject to boundary conditions, i.e. $f_{i,j}$ and $g_{i,j}$ are known if (im,jm) belong to the boundary. We assume that the surface normal at a boundary point (i,j) is parallel to the image plane (i.e. $f_{i,j}^2 + g_{i,j}^2 = 4$). (Occluding boundary).

If perspective projection is used, then the occluding boundary surface normals are not parallel to the image plane, *i.e.* $f^2 + g^2 \neq 0$, but they are computable. The boundary surface normals have only to be known, they don't have to have any particular value.

Function e of equation (15) is defined on a compact subset K of R^{2n}, with $n = k^2$, and it is continuous with respect to $f_{i,j}$ and $g_{i,j}$. Therefore, there exists a solution to the minimization problem. Furthermore, the solution that minimizes e, is the solution of the system

$$\frac{\partial e}{\partial f_{i,j}} = \frac{\partial e}{\partial g_{i,j}} = 0. \tag{17}$$

Equations (17) become

$$f_{i,j} = f_{i,j}^* - \frac{1}{4}\lambda m^2\Big[L(f_{i,j}, g_{i,j}, i,j)\Big]\frac{\partial L}{\partial f}(f_{i,j}, g_{i,j}, i,j)$$
$$g_{i,j} = g_{i,j}^* - \frac{1}{4}\lambda m^2\Big[L(f_{i,j}, g_{i,j}, i,j)\Big]\frac{\partial L}{\partial g}(f_{i,j}, g_{i,j}, i,j) \tag{18}$$

where

$$f_{i,j}^*(g_{i,j}^*) = \frac{f_{i+1,j}(g_{i+1,j}) + f_{i,j+1}(g_{i,j+1}) + f_{i-1,j}(g_{i-1,j}) + f_{i,j-1}(g_{i,j-1})}{4}$$

Equations (18) can be written as

$$\Phi\xi = -\lambda m^2\phi(\xi) \text{ where} \tag{19}$$

$$\xi = [f_{1,1}, \ldots, f_{1,k}, \ldots, f_{k,k}, g_{1,1}, \ldots, g_{kk}]^T$$

$$\phi = [\cdots, \{L(f_{i,j}, g_{i,j}, i,j)\}\frac{\partial L(f_{i,j}, g_{i,j}, i,j)}{\partial f}, \ldots, \{L(f_{i,j}, g_{i,j}, i,j)\}\frac{\partial L(f_{i,j}, g_{i,j}, i,j)}{\partial g}, \cdots]^T$$

and

$$\Phi = \begin{bmatrix} A & 0 \\ 0 & A \end{bmatrix} \text{ where } A \in \mathbf{R}^{n\times n} \text{ and}$$

$$A = \begin{bmatrix} B & -I & & & \\ -I & B & -I & & \\ & \cdots & & & \\ & & -I & B & I \\ & & & -I & B \end{bmatrix} \text{ and}$$

$$B = \begin{bmatrix} 4 & -1 & & & \\ -1 & 4 & -1 & & \\ & \cdots & & & \\ & & -1 & 4 & 1 \\ & & & -1 & 4 \end{bmatrix} \in \mathbf{R}^{k\times k}$$

Equation (19) is nothing but equations (18) written in a compact form. Notice that equation (19) is the necessary condition for the solution that minimizes (16). We will now prove that equation (19) has a unique solution. For this we will need the fact that the functions $[L(f,g,i,j)]\frac{\partial L(f,g,i,j)}{\partial f}$ and $[L(f,g,i,j)]\frac{\partial L(f,g,i,j)}{\partial g}$ are Lipschitz with respect to f and g.

Let ξ_1 and ξ_2 two solutions of equation (19), with $\xi_1 \neq \xi_2$. Then

$$\Phi\xi_1 = -\lambda m^2\phi(\xi_1) \tag{20}$$

$$\Phi\xi_2 = -\lambda m^2\phi(\xi_2). \tag{21}$$

But Φ is invertible [29]. So, (20) and (21) become:

$$\xi_1 = -\lambda m^2\Phi^{-1}\phi(\xi_1) \tag{22}$$

$$\xi_2 = -\lambda m^2\Phi^{-1}\phi(\xi_2). \tag{23}$$

From (22) and (23) we get

$$\xi_1 - \xi_2 = -\lambda m^2\Phi^{-1}\big(\phi(\xi_1) - \phi(\xi_2)\big),$$

$$\text{or } \|\xi_1 - \xi_2\|_2 \le \lambda m^2\|\Phi^{-1}\|_2\,\|\phi(\xi_1) - \phi(\xi_2)\|_2 \tag{24}$$

$$\text{But } \|\Phi^{-1}\|_2 = \left[8\sin^2\left(\frac{\pi m}{2}\right)\right]^{-1} < \left[2\pi^2m^2\left(1 - \frac{\pi^2m^2}{24}\right)^2\right]^{-1} \tag{25}$$

from [30].

Also, since $\left\{L(f,j,\ i,j)\right\}\dfrac{\partial L}{\partial f}$ and $\left\{L(f,g,\ i,j)\right\}\dfrac{\partial L}{\partial g}$ are Lipschitz with respect to f and g, we have

$$\left|\left\{L(f,g,\ i,j)\right\}\frac{\partial L(f,g,\ i,j)}{\partial f} - \left\{L(f',g',\ i,j)\right\}\frac{\partial L(f',g',\ i,j)}{\partial f}\right| \tag{26}$$

$$\le c_{i,j}\{(f - f')^2 + (g - g')^2\}^{1/2}$$

$$\left|\left\{L(f,g,\ i,j)\right\}\frac{\partial L(f,g,\ i,j)}{\partial g} - \left\{L(f',g',\ i,j)\right\}\frac{\partial L(f',g',\ i,j)}{\partial g}\right| \tag{27}$$

$$\le d_{i,j}\{(f - f')^2 + (g - g')^2\}^{1/2}$$

$$\text{and} \quad \mu = \max_{i,j}\{c_{i,j},\ d_{i,j}\} \tag{28}$$

From (26), (27), and (28) we get

$$\|\phi(\xi_1) - \phi(\xi_2)\|_2 \le \mu\|\xi_1 - \xi_2\|_2 \tag{29}$$

Equation (24) from (25) and (29) becomes

$$\|\xi_1 - \xi_2\|_2 < \lambda m^2\left[2\pi^2m^2\left(1 - \frac{\pi^2m^2}{24}\right)^2\right]^{-1}\mu\|\xi_1 - \xi_2\|_2 \tag{30}$$

If we choose λ such that

$$\lambda m^2\left[2\pi^2m^2(1 - \pi^2m^2/24)^2\right]^{-1}\mu < 1,$$

then equation (30) leads to a contradiction. So $\xi_1 = \xi_2$ and equation (19) has a unique solution. Furthermore, the sequence $\xi^{(\alpha)}$ defined by

$$\xi^{(\alpha+1)} = -\lambda m^2\Phi^{-1}\phi(\xi^{(\alpha)},\ \alpha = 0, 1, 2, \cdots$$

converges to the unique solution of equation (19). The proof proceeds as previously, by proving that if ξ is the solution, then $\|\xi^{(\alpha+1)} - \xi\|_2 < q\,\|\xi^{(\alpha)} - \xi\|_2$ where $q < 1$.

So, we have proved the following theorem.

Theorem:

If $0 < \lambda < 2\pi^2\mu^{-1}[1 - \pi^2m^2/24]^2$, then there exists a unique surface minimizing e (equation 16), which is also the unique solution of equation (19) and the above described algorithm converges to that solution.

To complete the proof we need to show that (26) and (27) hold. Observe that $L(f,g,i,j)$ is a polynomial in f,g. Consequently, $L\dfrac{\partial L}{\partial f}$ and $L\dfrac{\partial L}{\partial g}$ are also polynomials for every i,j. So, we can take the Lipschitz constant $c_{i,j}$ for (26) to be such that

$$c_{ij} \le \left\{\left[\sup_{f^2+g^2\le 4}\left|\frac{\partial}{\partial f}\left(L_{ij}\frac{\partial L_{ij}}{\partial f}\right)\right|\right]^2 + \left[\sup_{f^2+g^2\le 4}\left|\frac{\partial}{\partial g}\left(L_{ij}\frac{\partial L_{ij}}{\partial f}\right)\right|\right]^2\right\}^{\frac{1}{2}} \tag{31}$$

$$d_{ij} \le \left\{\left[\sup_{f^2+g^2\le 4}\left|\frac{\partial}{\partial f}\left(L_{ij}\frac{\partial L_{ij}}{\partial g}\right)\right|\right]^2 + \left[\sup_{f^2+g^2\le 4}\left|\frac{\partial}{\partial g}\left(L_{ij}\frac{\partial L_{ij}}{\partial g}\right)\right|\right]^2\right\}^{\frac{1}{2}} \tag{32}$$

(31) and (32) also mean that $c_{i,j}$ and $d_{i,j}$ exist and are finite for every i,j. Finiteness is also clear since the suprema are taken over the disc of radius 2 (compact set) and the functions are polynomials. In this context, if we replace $c_{i,j}$ and $d_{i,j}$ by their respective upper bounds of (31) and (32), it is possible that the resulting value of μ is larger than the actual one. However, this is immaterial for the uniqueness proof, since it only results in a smaller range for λ.

4.2. Summary

Up to now we have established the fact that from the knowledge of the surface normals at the boundaries, the constraint of the particular problem and the smoothness condition, we can uniquely recover the surface in view, and we have given an algorithm for this. The object in view had to have a square boundary but a generalization to any kind of boundary is possible. It has to be noted, though, that there are two important shortcomings in our theory up to now. The first has to do with the choice of the regularization parameter λ; λ can be chosen from an interval of values, but the solution we get might not be physically plausible. The second issue has to do with the fact that we used a specific smoothness condition, in particular we measured departure from smoothness with the functional

$$\int\int\left(f_x^2 + f_y^2 + g_x^2 + g_y^2\right)dxdy$$

The choice of this functional is clearly ad-hoc. The question that arises then is: can we build our theory in such a way so that we can learn the parameters that are involved in our problem and at the same time avoid any explicit smoothness constraints?

If we do not assume any particular variational condition for smoothness, as long as the condition is quadratic and if in equation (19) we hide the regularization parameter λ in matrix Φ, then from the minimization, we will always get an equation of the form

$$\Phi\xi = \phi(\xi)$$

where ξ is the shape vector, *i.e.* the surface normals in the image, Φ a matrix and ϕ a function of the shape that depends on the particular problem that we are analyzing. This is a nonlinear equation, for which we have proved that if some weak conditions on ϕ are satisfied and if matrix Φ is such that $\|\Phi^{-1}\|_2\cdot\mu<1$, where μ the max-

imum of the Lipschitz constants of the functions $\phi\frac{\partial\phi}{\partial f}$ and $\phi\frac{\partial\phi}{\partial g}$, then the iterative formula discussed in Section 4.1 will converge to a unique solution.

So, what we need to do is to present a way in which a neural net can learn matrix Φ from examples, in such a way that uniqueness, robustness and convergence are guaranteed. This, is discussed in the next section.

5. IMPLEMENTATION OF SHAPE COMPUTATION AND LEARNING

Techniques for learning the solutions to equations have been developed by researchers in the stochastic approximation [1,2], statistics [3, 7], mathematical learning theory [4, 5], computer science [6], and electrical engineering communities [8,9, 10]. Many of these techniques apply to linear as well as nonlinear equations. There are special characteristics that low-level vision problems have that may make it possible for us to accelerate the convergence of these techniques. This is because we sometimes have *a priori* knowledge that the solution must lie in a restricted subset of all possible solutions. We may also be obliged to find the solution within a constricted subspace, if we are to represent the solution by a neural network of limited size with a limited number of connections, which may not be able to represent an arbitrary function or even an arbitrary matrix. Techniques have been developed for finding the best solution within a restricted subspace.

First, we must discuss what exactly we want to learn and why. What does it mean to learn a solution to

$$\Phi\xi = \phi(\xi) \tag{33}$$

when there may be not a unique ξ that satisfies (33)? What does it mean to have learned Φ from examples when the examples might not suffice to determine Φ or when they give contradictory evidence as to what Φ is?

The Φ we choose is the least squares solution. Under reasonable and quite general statistical assumptions, the least squares solution is the most probable value for Φ. Computing the least squares solution involves calculating the Moore-Penrose inverse. In the following sections, we describe a recursive procedure for computing this inverse. This procedure is optimal in the sense that it produces as accurate an estimate of Φ as we can obtain from our limited set of examples, but it requires our neural net to store large matrices and make very complex calculations. So instead of searching for the optimal solution of (33) within a full n^2 dimensional vector space (n is the dimension of the shape vector), we calculate the best solution within a restricted subspace. The procedure for calculating this restricted subspace solution is very similar to that for calculating the full space solution but the computational complexity is much less and the speed of convergence is much greater. Another technique we discuss is the Robins-Monro procedure. This results in slower convergence but the calculations are very easy to implement on a neural net. Whatever procedure we use, it may take too many examples to learn Φ . We can use a priori knowledge to accelerate convergence. For example, we might know that Φ varies smoothly as a function of position. The Φ we use is partly determined by examples and partly by a priori knowledge. As we acquire more examples, the influence of a priori knowledge on our estimate of Φ decreases.

Before discussing these computations, let us review what learning (rather than stipulating a priori) the value of Φ will accomplish for us. Φ hides the regularization parameter λ. This parameter might be allowed to vary from place to place. (We might require a different amount of smoothing near the boundary than at the center of the visual field.) Our smoothing condition might involve first, second, or higher-order derivatives or some linear combination of derivatives of different orders. By varying Φ, we can take into account all these possibilities. We do not know which of these possibilities is correct.

That is why we want a general learning theory to help us find the proper Φ (to find the least squares solution to (33)).

5.1. The Generalized Inverse

As far as finding Φ is concerned, it is irrelevant that $\Phi(\xi)$ is a function of ξ. We just treat (33) as a linear equation to be solved for Φ. We are given a set of learning examples $(\xi_i , \phi(\xi_i))$ and we want to solve the system of equations

$$\Phi\ \xi_i = \phi(\xi_i) \tag{34}$$

Even if a teacher tells us the exact value of the shape vector, ξ_i, and of the image data that determine $\phi(\xi_i)$, we cannot expect (34) to be a strict equality because the model we used to determine Φ was an oversimplification. We cannot possibly take all factors into account. If the error is a zero-mean, normally distributed variable, it makes sense to seek a solution in the sense of least squares:

$$\text{minimize} \sum_i ||\Phi\ \xi_i - \phi(\xi_i)||_2^2 \tag{35}$$

Assuming normality of the error, then the least-squares solution to (59) is the most likely value for Φ. Even if the error $\Phi\xi_i - \phi(\xi_i)$ is not Gaussian we can under reasonable assumptions obtain a central limit theorem which says that if there are a large enough number of examples the least square solution will be the best.

In case there are many least-squares solutions, choose the Φ of minimum norm, $||\Phi||_2$. This will help to make Φ sparse. The minimum-norm, least squares solution to $AX = I$ is called the Moore-Penrose pseudo-inverse, A^+.

This inverse shares many properties with the ordinary inverse, for example:

$A^+ A$ and $A\ A^+$ are symmetric,

$A^+ A\ A^+ = A^+ ;\ A\ A^+ A = A.$

This inverse can be used to solve linear equations. The

least squares solution to AXB = C is $X = A^+ C B^+ + M(I - A^+ A M B B^+)$ where M is an arbitrary matrix of appropriate dimensions [7]. Other generalized inverses have been defined [12], but if $H^+ H = I$, then $H^+ Z$ is the only least squares solution to HX = Z. $H^+ H = I$ if and only if zero is the only null vector of H.

Let us apply this insight to (34). First note that if $\Phi_1, \ldots \Phi_n$ represent the rows of Φ, (34) is really a set of independent equations:

$$\text{for all i}, \Phi_k \xi_i = \phi_k(\xi_i) \tag{36}$$

where $\Phi_k(\xi_i)$ is the kth element of the vector $\Phi_k(\xi_i)$. For convenience, we can write this as

$$\xi_i{}^T \Phi_k{}^T = \Phi_k(\xi_i).$$

This has a unique least squares solution if the rank of the matrix of examples
$(\xi_1, \ldots, \xi_s)$ is at least n. So there should be at least n independant examples.

There are other reasons we will need at least n examples. We want our learning procedure to be robust. However, $A_j \rightarrow A$ does not imply $A_j{}^+ \rightarrow A^+$ unless $\lim(rank A_j) = rank(A)$ as j approaches infinity. We can guarantee this as long as we know we are only dealing with matrices of rank n. There are many methods for calculating generalized inverses. They all implicitly or explicitly make a decision about the rank of the matrix being inverted. Furthermore, we should remember we do not have an intrinsic need to know Φ. We want to be able to learn Φ so that given new image data, we can iteratively solve the equation

$$\xi = \Phi^+ \phi(\xi) \tag{37}$$

for ξ. We learn Φ in order to obtain an accurate value for ξ. Temporarily ignore the dependence of $\phi(\xi)$ on ξ and pretend the image data (or a teacher) told us the value of $\phi(\xi)$. We want an estimate of ξ that is linear in $\phi(\xi)$, that is unbiased, and that has minimum variance (a Best Linear Unbiased Estimate or BLUE). Whether (37) is a BLUE for ξ depends on the error covariance matrix

$E((\Phi_k \xi_i - \phi_k(\xi_i)) \cdot (\Phi_l \xi_j - \phi_l(\xi_j)))$ where E stands for expected value. However, if there are least n independent examples, we know that (37) is the statistically correct BLUE estimate.

5.2. Computing the Pseudo Inverse

Many techniques exist for computing pseudo-inverses; not all of them are suited to the case where the data arrive in a stream (a temporal succession of examples.) One method that is suitable is Greville's [7, 12].

The expression we want to compute is of the form

$$\Phi_{ks}{}^T = H_s{}^{T+} \phi_k(H_s) \tag{38}$$

where Φ_{ks} is our guess of row Φ_k after having seen s examples. H_s is the matrix formed from the first s examples of shape vectors $(\xi_1, \xi_2, \ldots, \xi_s)$ and $\phi_k(H_s)$ is the vector

$$[\phi_k(\xi_1)\ \phi_k(\xi_2) \cdots \phi_k(\xi_s)]^T$$

Initialize by using the estimate $\Phi_{k0}{}^T = 0$. Then we can use the recurrence equations

$$\Phi_{k(s+1)}{}^T = \Phi_{ks}{}^T + K_{s+1}\, error \tag{39}$$

where *error* is the error

$$\xi_{s+1} \Phi_{ks}{}^T - \phi_k(\xi_{s+1}) \tag{40}$$

obtained by using the s^{th} estimate, Φ_{ks}, on the s + 1 th example. The new estimate is obtained by adding to the old estimate a term proportional to the error. The gain matrix, K_{n+1} is given by:

$$K_{s+1} = \frac{(I - H_s{}^+ H_s)\xi_{s+1}^T}{\xi_{s+1}(I - H_s{}^+ H_s)\xi_{s+1}^T} \tag{41}$$

if $(I - H_s{}^+ H_s)\xi_{s+1} \neq 0$ and

$$K_{s+1} = \frac{H_s{}^+ H_s{}^{T+} \xi_{s+1}^T}{1 + \xi_{s+1} H_s{}^+ H_s{}^{T+} \xi_{s+1}^T} \tag{42}$$

otherwise.

(41) and (42) give the correct least squares solution, but depending on the hardware one has available, computing these equations on one's neural net may be a problem if one has to store the entire matrix of examples, H_s. The shape vectors are each of size n and one may have to store s of these. s might be much larger than n, and n itself can be large if one's grid is very fine and one has to store two parameters, f and g, for each point on the grid. There are several ways to deal with this problem. One is to recursively compute 2 auxillary matrices, $A_s = I - H_s{}^+ H_s$ and $B_s = (H_s{}^T H_s)^+$. The matrix K_{s+1} can be written in terms of A_{s+1} and B_{s+1}. These matrices are only of size n by n.

The recursive computations are [7]:

$$A_{s+1} = A_s - \frac{(A_s \xi_{s+1}^T)(A_s \xi_{s+1}^T)^T}{\xi_{s+1} A_s \xi_{s+1}^T} \tag{43}$$

if x_{s+1} is independent of the previous examples $\xi_1, \ldots, \xi_s$, and

$$A_{s+1} = A_s$$

otherwise. Similarly, we get equation (44) below:

$$B_{s+1} = B_s - \frac{(B_s \xi_{s+1}^T)(A_s \xi_{s+1}^T)^T + (A_s \xi_{s+1}^T)(B_s \xi_{s+1}^T)^T}{\xi_{s+1} A_s \xi_{s+1}^T} + \frac{1 + \xi_{s+1} B_s \xi_{s+1}^T}{(\xi_{s+1} A_s \xi_{s+1}^T)^2} (A_s \xi_{s+1}^T)(A_s \xi_{s+1}^T)^T$$

if ξ_{s+1} is independent of the previous examples and

$$B_{s+1} = B_s - \frac{B_s \xi_{s+1}^T (B_s \xi_{s+1}^T)^T}{1 + \xi_{s+1}^T B_s \xi_s}$$

otherwise.

We may still have an implementation problem because the two matrices A_s and B_s are of size n by n which is fairly large and the recursive computations (43) and (44) are fairly complex. When we have available to

us an arbitrary number of connections between an arbitrary number of nodes, there is no problem in computing these recursions in parallel in a small amount of time, but our resources may be limited. It is true that Φ is of size n by n and we do have to store Φ. But we usually have a priori knowledge that Φ is sparse. We can take advantage of this fact to simplify our computation.

We may not want to do that. One thing that does help us is that K_s does not depend on the data terms $\phi(\xi_i)$. So K_s does not depend on which particular "shape from" method we are using. If we know in advance what the set of example shape vectors $(\xi_1 \cdots \xi_s)$ will be, we can precompute the K_s and we might be able to choose our example shape vectors to simplify this calculation. Still, storing K_s might be a problem.

We can also restrict the solution space of possible values of Φ to those satisfying some additional constraint

$$G_k \Phi_k{}^T = U_k \tag{45}$$

for all k. Or more generally

$$\sum_k G_k \Phi_k{}^T = U \tag{46}$$

where G_k, U_k, and U depend on the constraint we want to impose. Learning within a constricted space of possible solutions is a case intermediate between the situation where we arbitrarily choose a particular smoothness measure in an entirely adhoc manner and the situation where we search the whole space of possible Φ's. We, for simplicity, will only deal with the constraint (45). (46) can be handled in a similar manner. Then our recursion takes the form.

$$\Phi_{k0}^T = G^+ U \tag{47}$$

$$\Phi_{ks+1}^T = \Phi_{ks}^T + \overline{K}_{s+1}(\overline{error}) \tag{48}$$

where

$$\overline{error} = (\overline{\phi_k(\xi_{s+1})} - \overline{\xi}_{s+1}\Phi_{ks}^T) \tag{49}$$

where

$$\overline{\phi_k(\xi_{s+1})} = \phi_k(\xi_{s+1}) - \xi_{s+1}G_k{}^+ U \tag{50}$$

and $\overline{\xi_{s+1}} = \xi_{s+1}^T(I - G^+G)^T$.

$$\overline{K_{s+1}} = \frac{\overline{A_s \xi_{s+1}}^T}{\overline{\xi_{s+1}}\,\overline{A_s \xi_{s+1}}} \tag{51}$$

if $\overline{\xi_{s+1}}$ is not a linear combination of $\overline{\xi_1}, \ldots, \overline{\xi_s}$, and

$$\overline{K_{s+1}} = \frac{\overline{B_s \xi_{s+1}}^T}{1 + \overline{\xi_{s+1}}^T \overline{B_s \xi_{s+1}}^T}$$

otherwise, where $\overline{A_s} = I - \overline{H_s}^+ \overline{H_s}$, $\overline{B_s} = (\overline{H_s}^T \overline{H_s})^+$ and $\overline{H_s} = H_s(I - G^+G)$.

The recursions for $\overline{A_s}$ and $\overline{B_s}$ are the same as those for A_s and B_s except that ξ_s is replaced by $\overline{\xi_s}$.

The equations (47)–(57), are the same as those in the unconstrained case except for the bars, which restrict our attention to the subspace such that $G_k(\Phi_k) = U_k$. If the null space of G_k has dimension D, then $I - G_k G$ has rank D and we confine our attention to a D-dimensional subspace. The matrices, $\overline{A_s}$ and $\overline{B_s}$ are only of size D^2.

Another approach is to change our optimality criterion [4]. Instead of learning the least squares solution to the whole system, we learn the Φ that produces the least error on the last r examples. Then we do not need to compute $H_s{}^+ H_s$, but only the matrix $H_s{}^{r+} H_s{}^r$ where $H_s{}^{rT} = (\xi_{s+1-r}, \ldots, \xi_{s+1})$. In the limit as s ->infinity, r > 1, the solution using this criterion will converge if we assume stationarity (the same Φ is valid for all times. Thus, there exists a Φ such that the model $\Phi\xi = \phi(\xi)$ plus noise holds for all examples where the noise is independent of ξ and the noise for the different examples is a set of statistically independent, identically distributed random variables. If the model is not stationary, the correct Φ changes in order to adopt to change in the world.

In the case r = 1, we will obtain the matrix $\xi_{s+1}(\xi_{s+1}{}^T \xi_{s+1})^{-1}$. To insure convergence, we change this to $\frac{1}{s+1}\xi_{s+1}(\xi_{s+1}{}^T \xi_{s+1})^{-1}$.

5.3. Suboptimal Gain Sequences

The methods we have discussed so far for computing least squares solutions involve computation of complex expressions in order to determine the matrices K_s. We can, however, work with much simpler matrices. For example, we can assume the matrix K_s is of the form [4]

$$K_s = a_s \xi_s \tag{52}$$

The a_s must satisfy the conditions

$$a_s > 0 \quad \sum a_s = \infty \quad \sum a_s{}^2 < \infty \tag{53}$$

The condition $\sum a_s = \infty$ insures that the later examples in the sequence ξ_i have sufficient influence on the matrix learned. The other condition causes convergence.

In this case, we have convergence with probability one to the correct least squares solution. The convergence rate is slower than optimal. This method for finding the solution to equations was originally developed by Robbins and Monro [13]. One can show that if one chooses $a_s = \frac{1}{s}$, then the convergence rate is asymptotically $O(\frac{1}{s})$ [14].

The Robbins-Monro procedure can be shown to converge with probability one to the least-squares solution to $\Phi\xi = \phi(\xi)$ under assumptions much weaker than the requirement that the errors be Gaussian. For example we can require only that the errors have identical, independent distributions and then we can obtain a convergence proof. Or we can allow the errors to be dependent on the state ξ because we really do not have the equation $\Phi\xi = \phi(\xi) + error$. Rather we have

$$\Phi(\xi + error_1) = \phi(\xi + error_2) + error_2 + error_3 \tag{54}$$

where $error_1$ takes into account the fact that we can never know ξ with total precision. It also allows for the

act that $\Phi\xi=\phi(\xi)$ is usually obtained by maximum likelihood reasoning. The ξ we find is the one that best fits (that is most likely) given the data but occasionally improbable events can occur. One cannot expect the actual ξ to be exactly equal to the most probable ξ. $Error_2$ arises because we do not know the data and hence we do not know ϕ with total precision. $Error_3$ is due to the factors our model fails to take into account. As long as we satisfy some simple conditions, we still get almost sure convergence to the least-squares solution. We can simply require that the errors in the different examples be bounded, almost surely have finite variance, be identically distributed and independent. These conditions are still somewhat restrictive. There are extensions to cases where the errors are a moving average of exogenous variables. For more copmplete discussions of when the Robbins-Monro procedure converges, see [1, 10, 5]. Note that the important assumption of bounded errors is reasonable if the derivatives of ϕ are bounded.

Even if we have convergence, we need not have convergence to the correct solution if our sample examples are not representative of the naturally occuring shapes. The solution will instead converge to the least-squares solution to the artificial problem presented by the atypical examples. What we require is that the vector spaces of naturally occuring possible examples be spanned infinitely often. Thus, for every m, the rank of the matrix $(\xi_m, \xi_{m+1}, \cdots, \xi_s, \cdots)$ must be n. And we also require that we can choose a function $f(i)$ such that for all small i, the shape vectors $\xi_{f(i)}, \cdots, \xi_{f(i+1)-1}$ span the vector space and $\sum \frac{1}{f(i)} = \infty$.

5.4. Acceleration of Convergence

Even if we use the optimal formula for estimation of the least squares solution, convergence can be quite slow. The variance of our estimates depends on the trace of the gain matrix. Thus, our rate of learning is inversely proportional to a large number n.

The result of this slow convergence is that our estimate Φ may not be as sparse as it should be. Another possibility is that some of the values of $\Phi = <\Phi_{ij}>$ might be misleading. We might have a reasonably accurate estimate of each Φ_{ij} but a bad estimate of the high frequency components such as $\Phi_{ij+1} - \Phi_{ij}$. The estimate of these components might simply reflect noise. This may mean that although we can compute shape accurately, we cannot accurately compute the difference in orientation of two nearby points.

In other words, we still have to deal with the regularization problem. We might make the ad-hoc assumption that $(\frac{\partial\Phi}{\partial i})^2 + (\frac{\partial\Phi}{\partial i})^2$ is small except at those i,j that reflect the information that data at a given point P gives us information about shape in the immediate vicinity of P. So, regarding Φ as a continuous function, we will have a condition of the form

$$\text{minimize} \frac{1}{n^2}\sum(\Phi_{ij} - \bar{\Phi}_{ij})^2 + \lambda\int_{\Omega}\left|(\frac{\partial\Phi}{\partial i})^2 + (\frac{\partial\Phi}{\partial j})^2\right| \quad (55)$$

where $\bar{\Phi}_{ij}$ is the value given by the recursive least-squares estimation of Φ. The ad-hocness here is not as bad as the ad-hocness in assuming that shape is smooth. Here we are only assuming that the function relating the shape to the data is smooth almost everywhere (it should be smooth in the region Ω which excludes the values of Φ_{ij} corresponding to information that the data at point P provide about the shape at point P). Furthermore as we learn more and more examples, we can let $\lambda \to 0$. So, the influence of the adhoc smoothing condition lessens. Of course, we could have chosen some other smoothing condition. As long as this other condition is quadratic, we will obtain a linear constraint in Φ. In general we will have a stabilizing family of smoothing conditions:

$$\text{minimize} \frac{1}{n^2}\sum(\Phi_{ij} - \bar{\Phi}_{ij})^2 + L_s(\Phi,\phi) \quad (56)$$

where the L_s are bilinear forms and $L_s \to 0$ (see Morozov [16]).

Do we want to learn L_s? That is do we want to learn the linear condition (56) gives rise to? The set of linear transformations from R^{n^2} to R^{n^2} has dimension n^2 by n^2. We do not really want to learn a n^2 by n^2 matrix, unless we can a priori constrain the matrix (the matrix in the linear condition corresponding to (56)) to a small subspace. The only reason we needed the additional constraint (56) is that we need a large number of examples to accurately learn the value of Φ particularly if the data are noisy, which will be true if the examples are given by nature. We have to make temporary ad-hoc assumptions somewhere. It is not clear where we should make them. One thing though, is clear. We should not make adhoc assumptions of smoothness of shape. We can eventually learn the right variational condition. It is safer to make the assumption in (56) because eventually $L_s \to 0$ as we accumulate more example.

The other way to handle slow learning of Φ is to restrict the subspace of possible solutions. We can still apply the same basic Robbins-Monro procedure. Except when computing the error we have to use a formula like (49), and we project Φ_{s+1} into the restricted subspace of possible solutions. We can also restrict the solution to a closed convex subset and impose inequality constraints similar to the one that guarantees convergence, namely

$$||\Phi^+||_2 \cdot \mu < 1 \quad (57)$$

5.5. Varying Parameters to Obtain Unique Convergence

After we have learned Φ, our task is not necessarily done. If $||\Phi^+||_2 \cdot \mu < 1$ (58), where μ is the maximum of the Lipschitz constants, we have a unique fixed point to $\xi = \Phi^+\phi(\xi)$, which we compute iteratively. If $||\Phi^+||_2 \cdot \mu \geq 1$, it is not clear how to proceed, because convergence is not guaranteed. Of course, it may be the case that there is no unique fixed point. Another possibility is that $\phi(\xi)$ is the wrong function to use and that a mathematically equivalent formulation will satisfy (84).

For example if $\phi(\xi) = \Phi\xi$ we could use instead the constraint

$$(\Phi + I)\xi = \phi(\xi) + \xi$$

Or we may make a nonlinear transformation. For example, in one dimension a constraint of the form $x^4 + ax^3 + bx^2 + cx + d = x$ could be written as $x = (ax^3 + bx^2 + cx + d - x)^{\frac{1}{4}}$. There is an infinite dimensional space of possible ways of rewriting the constraint $\Phi\xi = \phi(\xi)$.It is not clear how to systematically find a reformulation for which (58) is true. The problem is that (58) must hold regardless of the data.

One way to get around this problem is to let Φ be a function of the data. Letting Φ be an arbitrary function of the data will not work. There is an easy trivial solution to $\Phi\xi = \phi(\xi)$. What we might do is divide the space of possible data into a small number of regions and then for each region R_i, learn the least-squares solution to the linear equation

$$\Phi_i\xi = \phi(\xi)$$

when the data is in R_i. The more regions R_i, we have, the longer it takes to learn all the Φ_i. So, we might stop subdividing R_i as soon as we find that $||\Phi_i^+||_2\mu < 1$, where the Lipschitz constant μ is only evaluated for ξ corresponding to data in R_i. We can as before impose a regularization condition, requiring smoothness accross boundaries. Future research is needed on how to let Φ vary with the data (what regions to choose and how to find mathematically equivalent formulations of (33)).

6. CONCLUSIONS

Many low-level vision problems have a common characteristic. We are given image data that do not uniquely determine the physical parameters of the object in view (as such parameters we chose shape in this paper). This problem is usually addressed by adding smoothness constraints that force the solution to be unique. We have shown how any quadratic smoothness constraint can be learned from examples by using well known mathematical techniques such as Greville's method for recursive computation of a least squares solution and the Robbins-Monro technique of stochastic approximation. The Robbins-Monro technique works under a variety of noise conditions and is easily implementable in massively parallel networks. This technique provides a uniform theory for learning smoothness constraints in low-level vision. The convergence can be accelerated by using a priori knowledge of the relationship between image data nad physical properties but the ultimate answer provided by the system does not depend on these assumptions. Once the connectionist network has learned all the parameters involved in the computation of real world properties from image data, it can use a simple iterative technique based on the theory of fixed points to find a unique solution to problems such as "shape from X". This fixed point method extends the usual linear regularization theory and shows how one can uniquely solve a non-linear variational problem provided the solution at the boundaries is known. We are currently experimenting with our theory, using synthetic and real images. The details of our connectionist implementation that is based on ideas from [38], and our experimental results will appear elsewhere.

REFERENCES

1: H. Kusher and D. Clark, *Stochastic approximation methods for constrained and unconstrained systems*, Springer Verlag, 1978.

2: M.T. Wasan, *Stochastic Approximation*, Cambridge, 1969.

3: L. Bromberg, *Bayesian analysis of linear models*, Marcel Dekker, 1985.

4: Y.Z. Tsypkin, *Foundation of the Theory of Learning Systems*, Academic Press, 1973.

5: U. Herkenrath, D. Kalin and W. Vogel (Eds.), *Mathematical Learning Models: Theory and Applications*, Springer 1983.

6: T. Kohonen, *Associative Memory: A Systems Theoretical Approach*, Springer, 1977.

7: A. Albert, *Regression and the Moore-Penrose Inverse*, Academic Press, 1972.

8: R.E. Kalman, "A new approach to linear filters and prediction problems", in *J. Basic Eng. Trans. ASME* series D., **82**, 35-45.

9: L. Ljung, "Analysis of recursive stochastic algorithms", *IEEE Trans. Aut. Control*, **AC-22**, 551-75, 1977.

10: P. Maybeck, *Stochastic Models, Estimation and Control*, Academic Press, 1979.

11: R. Penrose, "A generalized inverse for matrices", *Proc. Cambridge Phil. Soc.* **51**, 406-13, 1955.

12: A. Ben-Israel and T. Greville, *Generalized Inverses, Theory and Applications*, Wiley, 1974.

13: H. Robbins and S. Monro, "A stochastic approximation method", *Ann. Math. Stat.* **22**, 1951.

14: K.L. Chung, "On a stochastic approximation method", *Ann. Math. Stat.* **25**, 1954.

15: A.E. Albert and L.A. Gardner, *Stochastic Approximation and Nonlinear Regression*, MIT Press, 1967.

16: V.A. Morozov, *Methods for Solving Incorrectly Posed Problems*, Springer 1984.

16: B.K.P. Horn, *Robot Vision*, McGraw-Hill, 1986.

17: A. Witkin, "Recovering surface orientation and shape from texture", *Artificial Intelligence*, **17**, 1981.

18: J. Aloimonos, "Shape from texture", *Proc. IEEE CVPR*, 1986.

19: K. Stevens, "Shape from texture and contour", Ph.D. thesis, MIT, 1981.

20: S. Ullman, "The interpretation of structure from motion", *Proc.,R. Soc. Lond.* **B 203**, 405-426, 1979.

21: T. Kanade, "Determining the shape of an object from a single view", *Artificial Intelligence*, **17**, 1981.

22: J. Aloimonos and A. Basu, "Shape and motion from contour without correspondence", *Proc. IEEE CVPR*, 1986.

23: D. Marr, *Vision*, McGraw Hill, 1982.

24: A.N. Tichonov, and V.Y. Arsenin, *Solution of Ill-Posed Problems*, Winston and Wiley Publishers, Washington D.C., 1977.

25: T. Poggio and C. Koch, "Ill-posed problems in early vision: from computational theory to analog networks", *Proc. Royal Society of Lond.*,**B**, 1985.

26: S. Shafer, personal communication, 1986.

27: J. Aloimonos, "Computing intrinsic images", Ph.D. thesis, University of Rochester, 1986.

28: T. Poggio and V. Torre, "Ill-posed problems and regularization in early vision", *Artificial Intelligence Lab Memo*, no. 773, MIT, Cambridge, Massachusetts.

29: D. Lee, "An algorithm for shape from shading", Proc. Image Understanding Workshop, 1985.

30: G.D. Smith, *Numerical Solution of Partial Differential Equations: Finite Difference Methods*, Oxford University Press, 1978.

31: K. Ikeuchi, "Shape from regular patterns", *Artficial Intelligence*, **22**, 49-75, 1984.

32: J. Kender, "Shape from texture", Ph.D. thesis, Carnegie Mellon University, 1981.

33: T. Kanade and J. Kender, "Skewed symmetry: mapping image regularities into shape", Technical Report, CMU-CS-80-133, Dept. of Computer Science, Carnegie-Mellon University.

34: Y. Ohta, K. Maenobu and T. Sakai, "Obtaining surface orientation from texels under perspective projection", *Proc. 7th IJCAI*, Vancouver Canada, 1981.

35: J. Aloimonos and M. Swain, "Shape from texture", *Proc. IJCAI*, Los Angeles, CA, 1985.

36: J. Hadamard, *Lectures on the Cauchy Problem in Linear Partial Differential Equations*, New Haven: Yale University Press.

37: T. Poggio, "MIT Progress in understanding images", Proc. Image Understanding Workshop, 1985.

38: J. Feldman, "Four frames suffice: A provisional model of vision and space", *Behavioral and Brain Sciences*, June 1985.

LOCAL SHAPE FROM SPECULARITY

Glenn Healey and Thomas O. Binford

Artificial Intelligence Laboratory
Computer Science Department
Stanford University
Stanford, California 94305

Abstract

We show that highlights in images of objects with specularly reflecting surfaces provide significant information about the surfaces which generate them. A brief survey is given of specular reflectance models which have been used in computer vision and graphics. For our work, we adopt the Torrance-Sparrow specular model which, unlike most previous models, considers the underlying physics of specular reflection from rough surfaces. From this model we derive powerful relationships between the properties of a specular feature in an image and local properties of the corresponding surface. We show how this analysis can be used for both prediction and interpretation in a vision system. A shape from specularity system has been implemented to test our approach. The performance of the system is demonstrated by careful experiments with specularly reflecting objects.

1. Introduction

When light is incident on a surface, some fraction of it is reflected. A perfectly smooth surface reflects light only in the direction such that the angle of incidence equals the angle of reflection. For rougher surfaces, e.g. the surface of a metal fork, specular effects are still observable. In this paper we analyze the properties of specular reflection from rough surfaces.

There are numerous reasons why the study of specular reflection deserves serious attention in computer vision. Specular features are almost always the brightest regions in an image. Contrast is often large across specularities; they are very prominent. In addition, the presence or absence of specular features provides immediate constraints on the positions of the viewer and light sources relative to the specular surface. Also, as we will show, the properties of a specularity constrain the local shape and orientation of the specular surface.

An ability to understand specular features is valuable for any vision system which must interpret images of glossy surfaces. This work, motivated by experience with ACRONYM [4], began in order to provide the SUCCESSOR system with the capability to reason about specular reflection from metal parts in the IUA project [6]. Images of these parts typically contain large specular regions (Figure 1).

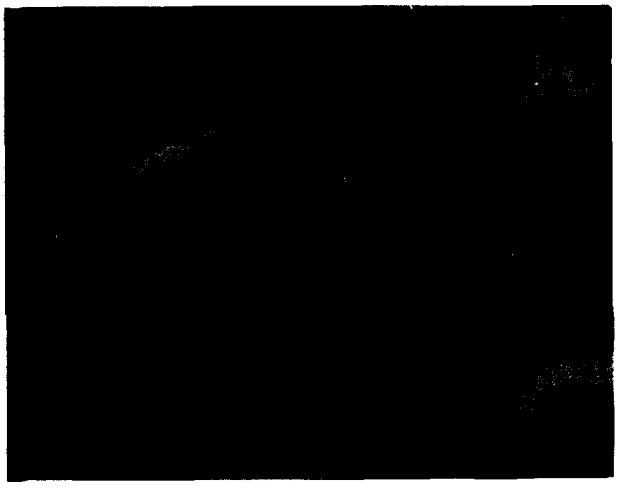


Figure 1. Typical Image Containing Specularities

We examine what information can be inferred from an image of a rough surface by considering the physics of specular reflection. Particular emphasis is placed on finding symbolic quasi-invariant relationships which will hold in many different situations (e.g. different source, viewer configurations). In contrast to many intensity-based vision algorithms, we compute a small number of local surface statistics based on the properties of a relatively large number of pixels in an image. This allows us to observe predicted features and infer local surface shape in noisy intensity images and in cases where available specular models do not completely characterize the physics of specular reflection.

2. Review of Previous Work

Researchers in computer graphics have used increasingly realistic specular models. Several of these models will be discussed in the next section. In computer vision, however, relatively few attempts have been made to exploit the information encoded in specularities. Ikeuchi [16] employs the photometric stereo method [24] and uses distributed light sources to determine the orientation of patches on a surface. Grimson [11] uses Phong's specular model [18] to examine specularities from two views in order to improve the performance of surface interpolation. Coleman and Jain [7] use four-source photometric stereo to identify and correct for specular reflection components. In more recent work, Blake [2] assumes smooth surfaces and single point specularities to derive equations to infer surface shape using specular stereo. He shows that the same equations can be used to predict the appearance of a specularity on a smooth surface when using a distributed light source. Takai, Kimura, and Sata [22] describe a model-based vision system which recognizes objects by predicting specular regions. As specular models and insights improve, we expect to see more work which makes use of the properties of specular reflection.

3. Specular Reflectance Models

Given a viewer, a surface patch, and a light source, a reflectance model quantifies the intensity the viewer will perceive. General reflectance models represent the perceived intensity I as a sum of two reflection components

$$I = I_D + I_S \qquad (1).$$

I_D represents the intensity of diffusely reflected light and I_S represents the intensity of specularly reflected light. In this paper we restrict our attention to the I_S reflection component.

We note that it is typically easy to separate the I_S reflection component from the I_D reflection component in an image. There are several distinctive properties of specular reflection. Over most of a surface I_S is zero, but in specular regions I_S is usually very large relative to I_D. In regions where the specular component is nonzero, I_S changes much more rapidly with surface geometry than I_D. Furthermore, the color of the I_S reflection component is almost always different from the color of the I_D reflection component.

Before discussing the various specular reflectance models, we introduce the reflection geometry (Figure 2). We consider a viewer looking at a surface point P which is illuminated by a point light source. Define

$\vec{V}$ = unit vector from P in direction of viewer
$\vec{N}$ = unit surface normal at P
$\vec{L}$ = unit vector from P in direction of source
$\vec{H} = \frac{\vec{V}+\vec{L}}{\|\vec{V}+\vec{L}\|}$ (unit angular bisector of $\vec{V}$ and $\vec{L}$)
$\alpha = \cos^{-1}(\vec{N} \cdot \vec{H})$ (the angle between $\vec{N}$ and $\vec{H}$)

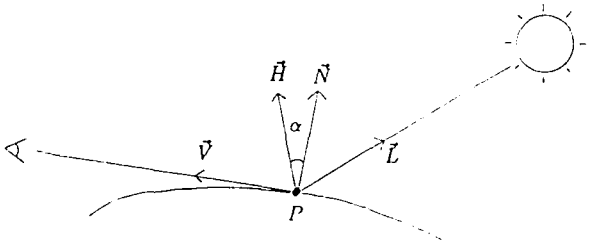


Figure 2. The Reflection Geometry

In describing specular models, we consider illumination from a single point light source. In principle, we lose no generality using this approach. In situations involving distributed light sources, we only need to integrate the effects of an equivalent array of point sources. A discussion of the geometry of extended sources is given in [14].

The simplest specular model assumes that specularities only occur where the angle of incidence equals the angle of reflection and $\vec{L}$, $\vec{N}$, and $\vec{V}$ all lie in the same plane. This corresponds to the situation $\alpha = 0$ in Figure 2. Unless the surface is locally flat, this model predicts that specularities will only be observed at isolated points on a surface. A few experiments, however, show that this model is inadequate for most real surfaces. Not only are observed specular features usually larger than single points, but highlights often occur in places which are not predicted by this model.

An empirical model for specular reflection has been developed by Phong [18] for computer graphics. This model represents the specular component of reflection by powers of the cosine of the angle between the perfect specular direction and the line of sight. Thus, Phong's model is capable of predicting specularities which extend beyond a single point. While Phong's model gives a reasonable approximation which is useful in some contexts, the parameters of this model have no physical meaning. It is possible to develop more accurate models by examining the physics underlying specular reflection.

The Torrance-Sparrow model [23], developed by physicists, is a more refined model of specular reflection. This model assumes that a surface is composed

of small, randomly oriented, mirror-like facets. Only facets with a normal in the direction of $\vec{H}$ contribute to I_S. The model also quantifies the shadowing and masking of facets by adjacent facets using a geometrical attenuation factor. The resulting specular model is

$$I_S = FDA \qquad (2)$$

where

F = Fresnel coefficient
D = facet orientation distribution function
A = adjusted geometrical attenuation factor

We will analyze the effects of each factor in the model in the next few paragraphs. The results we present in this paper are derived from equation (2).

The Fresnel coefficient F models the amount of light which is reflected from individual facets. In general, F depends on the incidence angle and the complex index of refraction of the reflecting material. Cook and Torrance [8] have shown that to synthesize realistic images, F must characterize the color of the specularity. The Fresnel equations predict that F is a nearly constant function of incidence angle for the class of materials with a large extinction coefficient [21]. This class of materials includes all metals and many other materials with a significant specular reflection component.

The distribution function D describes the orientation of the micro facets relative to the average surface normal $\vec{N}$. Blinn [3] and Cook and Torrance [8] discuss various distribution functions. All of these functions are very similar in shape. In agreement with Torrance and Sparrow we use the Gaussian distribution function given by

$$D = Ke^{-(\alpha/m)^2} \qquad (3)$$

where K is a normalization constant. Thus, for a given α, D is proportional to the fraction of facets oriented in the direction $\vec{H}$. The constant m indicates surface roughness and is proportional to the standard deviation of the Gaussian. Small values of m describe smooth surfaces for which most of the specular reflection is concentrated in a single direction. Large values of m are used to describe rougher surfaces with larger differences in orientation between nearby facets. These rough surfaces produce specularities which appear spread out on the reflecting surface. Figure 3 shows the effect of different values of m.

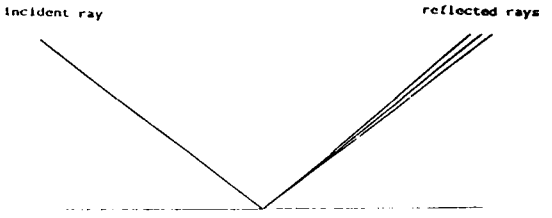


Figure 3a. Specular Distribution for Small m

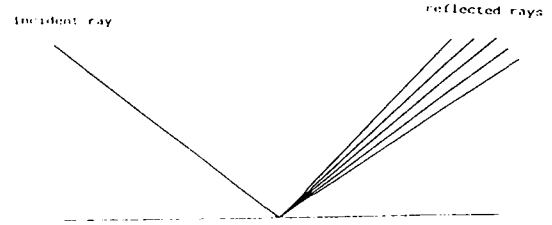


Figure 3b. Specular Distribution for Large m

The factor A quantifies the effects of a geometrical attenuation factor G corrected for foreshortening by dividing by $(\vec{N} \cdot \vec{V})$.

$$A = \frac{G}{\vec{N} \cdot \vec{V}} \qquad (4)$$

G is derived by Torrance and Sparrow in [23]. They assume that each specular facet makes up one side of a symmetric v-groove cavity. From this assumption, they examine the various possible facet configurations which correspond to shadowing or masking. The expression is

$$G = min\left\{1, \frac{2(\vec{N} \cdot \vec{H})(\vec{N} \cdot \vec{V})}{(\vec{V} \cdot \vec{H})}, \frac{2(\vec{N} \cdot \vec{H})(\vec{N} \cdot \vec{L})}{(\vec{V} \cdot \vec{H})}\right\} \qquad (5).$$

We will show that in applications it is often possible to use a simpler expression for G.

Let μ be the angle between $\vec{N}$ and $\vec{V}$. As μ increases from 0 to $\frac{\pi}{2}$, the viewer gradually sees a larger part of the reflecting surface in a unit area in the view plane. Therefore, as μ gets larger, there are correspondingly more surface facets which contribute to the intensity perceived by the viewer. We take this phenomenon into account in (4) by dividing by $\vec{N} \cdot \vec{V}$.

4. Shape from Specularity

In this section, we demonstrate how we can use (2) to determine local surface properties from specularities. In almost all situations we do not require the full generality of (2) to infer these local properties. Our first assumption is that F is a constant with respect to viewing geometry. This is a very good approximation for metals and for many other materials. We can further simplify (2) by observing that the exponential factor in (3) changes much faster than any of the terms of A. Therefore, except for a small range of angles near grazing incidence, A can be considered constant across the specularity. We will discuss the consequences of this assumption later. Hence, the form of (2) used to determine local surface properties is

$$I_S = K' e^{-(\alpha/m)^2} \qquad (6),$$

where K' is a constant.

Referring again to the geometry of Figure 2, we assume that the viewer and light source are distant relative to the dimensions of the surface. Therefore $\vec{V}$ and $\vec{I}$ may be regarded as constant; hence their angular bisector $\vec{H}$ is also constant. We assume that the positions of the viewer and light source are known. Finally, since the distance from the viewer to the surface is large, we can approximate the perspective projection of the imaging device with an orthographic projection.

4.1. Inferring Local Surface Shape

For a surface M on which the Gaussian curvature is locally nonzero, we will be able to locate a single point P_0 of maximum intensity in the image of the specularity. From (6) we see that this point corresponds to the local surface orientation $\vec{N} = \vec{H}$ (i.e. $\alpha = 0$). Given such a surface where $\vec{H}$ is known, we can immediately determine the surface orientation at P_0.

Figure 4 shows a typical intensity surface for a specular image. The level sets are image curves of constant specular intensity. P_0 corresponds to $\alpha = 0$. As predicted by (6), specular intensity decreases as we move away from P_0.

After locating P_0, we can transform the specular intensity image to the α angle image. Consider equation (6). If I' is the specular intensity corresponding to an arbitrary image point P' near P_0, then the angle α at the surface point imaging to P' is given by

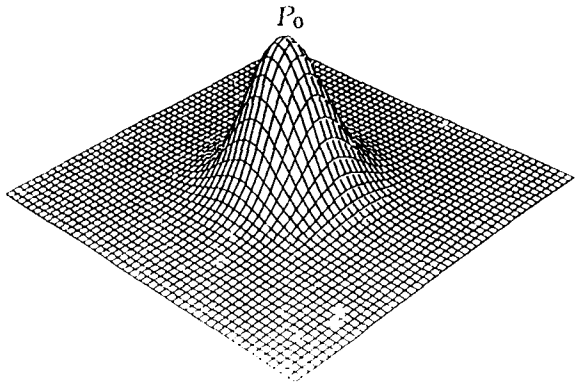


Figure 4. Specular Intensity Surface for a Curved Surface

$$|\alpha| = m\sqrt{-\ln \frac{I'}{K'}} \qquad (7)$$

We see that α is determined only up to sign. This will cause the sign of the normal curvature computed at P_0 to be ambiguous. In applications, this ambiguity can usually be resolved by considering other cues. From (7) we can compute the absolute value of α corresponding to each point P' in a neighborhood of P_0. The image of $|\alpha|$ values is called the α angle image.

From the α angle image, we can compute local curvature properties of the surface. Let $T_{P_0}(M)$ be the tangent space to M at P_0. To compute curvatures, we take a finite number n of straight line samples of the α angle image intersecting P_0. To insure uniform angular resolution on the surface, these samples must be taken in equally spaced directions in $T_{P_0}(M)$. In general, equally spaced directions in the image will not correspond to equally spaced directions in the tangent space. Thus, given a direction in the tangent space to the surface at P_0, we need to determine the corresponding direction in the image.

Consider a 3-D coordinate system such that the viewer is looking down the z-axis and P_0 is at the origin (Figure 5).

At P we have $\vec{N} = \vec{H}$ so that the vector $\vec{H}$ is normal to the tangent space to the surface at P. We choose to define angles in the tangent space in the counterclockwise

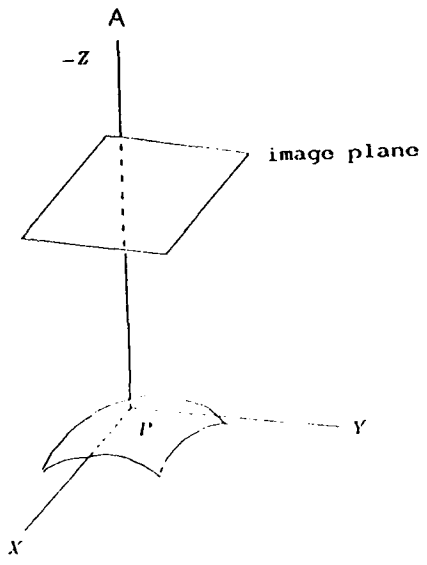


Figure 5. The Projection Geometry

sense from y = 0 and in the half space x ≥ 0. Denote the normal to the surface at P by $\vec{N} = (\vec{N}_1, \vec{N}_2, \vec{N}_3)$. The tangent space to the surface at P is given by

$$\vec{N}_1 x + \vec{N}_2 y + \vec{N}_3 z = 0 \quad (8)$$

Along y = 0, the unit vector V_0 in the tangent space is

$$V_0 = \left(\frac{-\vec{N}_3}{\sqrt{\vec{N}_3^2 + \vec{N}_1^2}}, 0, \frac{\vec{N}_1}{\sqrt{\vec{N}_3^2 + \vec{N}_1^2}} \right) \quad (9)$$

To simplify the notation let

$$K_1 = \frac{-\vec{N}_3}{\sqrt{\vec{N}_3^2 + \vec{N}_1^2}}, \quad K_2 = \frac{\vec{N}_1}{\sqrt{\vec{N}_3^2 + \vec{N}_1^2}} \quad (10)$$

Let θ be the angle of interest in the tangent space. The goal is to find a unit vector $\vec{V}_1 = (x_1, y_1, z_1)$ which lies in the tangent space and makes an angle θ with V_0. The angle θ provides the constraint

$$x_1 K_1 + z_1 K_2 = cos\theta \quad (11),$$

From (8) we must require

$$x_1 \vec{N}_1 + y_1 \vec{N}_2 + z_1 \vec{N}_3 = 0 \quad (12),$$

and since V_1 is a unit vector we have

$$x_1^2 + y_1^2 + z_1^2 = 1 \quad (13).$$

The equations (11), (12), (13) may be solved uniquely for x_1, y_1, z_1 in the half space x ≥ 0. Briefly, the solution is

$$z_1 = \frac{-R_2 + \sqrt{R_2^2 - 4R_1R_3}}{2R_1} \quad (14)$$

$$x_1 = C_4 - C_5 z_1 \quad (15)$$

$$y_1 = \frac{-x_1 \vec{N}_1}{\vec{N}_2} - \frac{z_1 \vec{N}_3}{\vec{N}_2} \quad (16)$$

where

$$R_1 = C_1 C_5^2 + C_2 \quad C_3 C_5 \quad (17a)$$

$$R_2 = -2C_1 C_4 C_5 + C_3 C_4 \quad (17b)$$

$$R_3 = C_1 C_4^2 - 1 \quad (17c)$$

$$C_1 = 1 + \frac{\vec{N}_1^2}{\vec{N}_2^2} \quad (18a)$$

$$C_2 = 1 + \frac{\vec{N}_3^2}{\vec{N}_2^2} \quad (18b)$$

$$C_3 = \frac{2\vec{N}_1 \vec{N}_3}{\vec{N}_2^2} \quad (18c)$$

$$C_4 = \frac{cos\theta}{K_1} \quad (18d)$$

$$C_5 = \frac{K_2}{K_1} \quad (18e)$$

A special case occurs when $\vec{N}_2 = 0$. For this case we use (11) - (13) to arrive at

$$z_1 = \frac{-\vec{N}_1 cos\theta}{K_1 \vec{N}_3 - K_2 \vec{N}_2} \quad (19)$$

$$x_1 = \frac{cos\theta - z_1 K_2}{K_1} \qquad (20)$$

$$y_1 = \sqrt{(1 - z_1^2 - x_1^2)} \qquad (21)$$

We use (14) - (21) to compute the components of V_1.

Assuming an orthographic projection and examining the geometry of Figure 5, we see that the x_1, y_1 components of V_1 give us the projected vector we are seeking in the image. Let θ' be the image angle corresponding to V_1. Then

$$\theta' = tan^{-1}\left(\frac{y_1}{x_1}\right) \qquad (22)$$

The next step is to use the α image to compute the normal curvature of the surface in the direction θ. The normal curvature is computed by taking a straight line L in the α image which intersects P_0 and is in the direction θ'. Under orthographic projection, L will project to a line L' in $T_{P_0}(M)$. The goal is to compute the normal curvature of the curve C $\subset$ M where C is the orthogonal projection of L' onto M. Since C projects to a line in $T_{P_0}(M)$, the magnitude of the geodesic curvature $|K_g|$ of C is 0 [17]. Thus local changes in α along C are due primarily to normal curvature along C. We compute the normal curvature κ_n in the direction θ by

$$\kappa_n = \left.\frac{d\alpha}{ds}\right|_{P_0} \qquad (23)$$

where s is arc length in the direction θ. In other words, we are differentiating the α image along L with respect to arc length on the surface. From the local character of specularities, we see that, to a very good approximation, arc length on the surface is equal to length in $T_{P_0}(M)$. Therefore, length in the image and arc length on the surface are related by the scale factor $\sqrt{x_1^2 + y_1^2}$. Thus, computing normal curvature on the surface has been *reduced to differentiation* in the α image.

If we let θ vary in the range $0 < \theta \leq 2\pi$ we can compute κ_n in any number of directions at P_0. The principal curvatures of M at P_0 are defined to be the maximum and minimum values of κ_n; the corresponding directions are called the principal directions. Hence, using this technique it is possible to describe M locally to second order in terms of principal curvatures and principal directions. In the context of shape from shading [3], Bruss [5] and Deift and Sylvester [9] examine the assumptions required to generate higher order surface descriptions from an α image.

4.2. Special Cases

In this subsection we examine specular reflection from special classes of surfaces. In 4.2.1. and 4.2.2. we consider surfaces which are locally singly curved and planar respectively. For these surfaces, the Gaussian curvature is locally zero. In 4.2.3. we examine the case of corners and edges where surface normal is discontinuous but where specularities are frequently observed.

4.2.1.Singly Curved Surfaces

If one principal curvature of a surface is zero in a specular region (i.e. the surface is locally singly curved), we will not be able to infer immediately the local orientation as we did for a doubly curved surface. To understand why, consider Figure 6. Figure 6 shows a viewer looking at a tilted cylinder. To make the example concrete, assume that $\hat{L}$ is such that $\hat{H} = \hat{V}$. For this configuration there will be no point on the surface for which $\alpha = 0$ (recall that $\hat{H}$ is essentially constant), yet we will still observe a specularity in the image if at some point α is small enough to give a significant value for I_S in (6). Define ϕ to be the smallest value of α for a given surface-source-viewer configuration. Figure 7 shows a specularity generated by a cylinder which is oriented so that ϕ is 20°. Note that a specular model which assumes a smooth surface would not predict a specularity for this case.

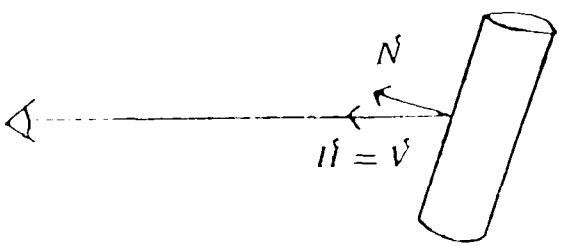


Figure 6. Viewer Observing a Singly Curved Surface

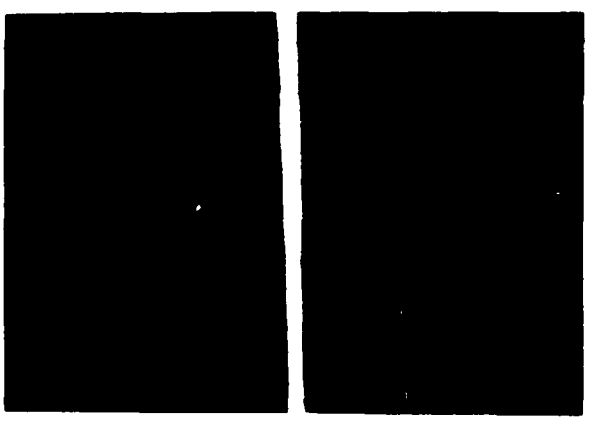


Figure 7. Specularity for a Tilted Cylinder

We observe that it is typically easy to detect that a surface is singly curved at a specularity. This is because we will observe a line of maximum intensity (along the line of zero curvature) instead of the point maximum we observe for the doubly curved case.

Figure 8 is a plot of I_S for a singly curved surface in a direction perpendicular to the lines of zero curvature as we change ϕ. It is worth noting that both the magnitude and shape of I_S change as ϕ increases. Consequently, it is possible to recover significant local shape information for this class of surfaces.

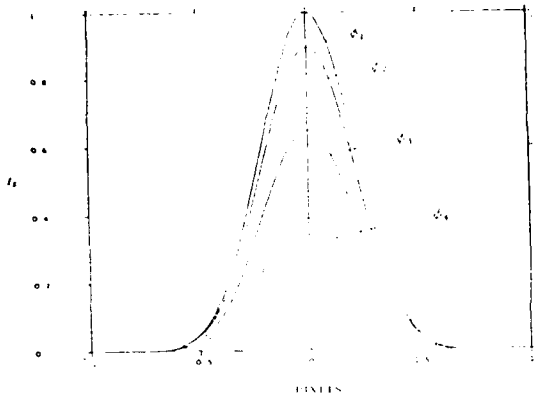


Figure 8. I_S for different values of ϕ

4.2.2. Planes

For a planar surface, $\hat{N}$ is constant. Hence, recalling our basic assumptions, I_S is constant across a plane. If the plane is oriented such that α is small enough, then a viewer will perceive an I_S reflection component. As with the singly curved surface, the magnitude of the perceived intensity will depend on α. If α is not sufficiently small, then I_S will be zero at all points on the plane. These observations provide us with two useful pieces of information:

1. Glossy surfaces which don't generate specularities over a range of orientations are probably planar.
2. Surfaces which produce a specularity of constant intensity over a 2-D region in the image are locally planar.

4.2.3. Corners and Edges

Specularities are often observed at places of discontinuous surface normal on an object. Typical examples of these discontinuities are edges and corners on a polyhedron. For an ideal edge on a polyhedron, the surface normal is discontinuous across the edge (Figure 9).

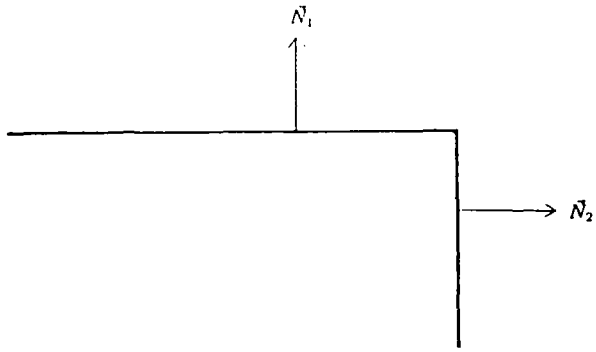


Figure 9. An Ideal Edge

For an ideal edge joining two planes, we should not expect to observe a specularity unless either $\hat{N}_1$ or $\hat{N}_2$ is oriented in a direction which is sufficiently close to the perfect specular direction $\hat{H}$. But for this case, as discussed in 4.2.2., we would expect to observe a spread out specular feature on one of the two planes joined by the edge. So why do we frequently see specular reflections along edges? On real polyhedra, surface normals are usually continuous across edges. Instead of the normal vector changing discontinuously, the normal usually changes smoothly from $\hat{N}_1$ to $\hat{N}_2$ by taking values which are linear combinations of $\hat{N}_1$ and $\hat{N}_2$. As we cross an edge, the surface normal moves rapidly through a large range of angles. If any of these normals is oriented in a direction sufficiently near the direction $\hat{H}$, we will observe a specularity. Therefore, we often observe a specularity on an edge. Figure 10 shows an image of an edge specularity.

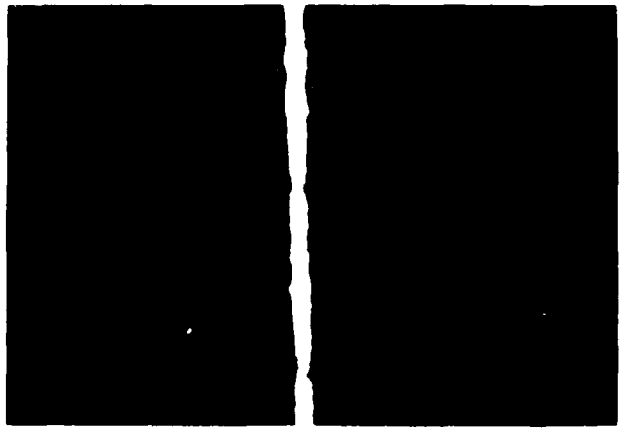


Figure 10. Image of an Edge Specularity

The situation is similar for trihedral vertices. As with edges, the normal vector is usually continuous at a cor-

ner. For trihedral vertices, the normal vector typically takes on values which are linear combinations of the three normals corresponding to the three planes defining the vertex.

From experiments with polyhedra, we have developed a useful model for the behavior of surface normal across edges. Define r to be the edge sharpness parameter and assume the coordinate system of Figure 11. P_1 and P_2 denote two planes intersecting to form an edge.

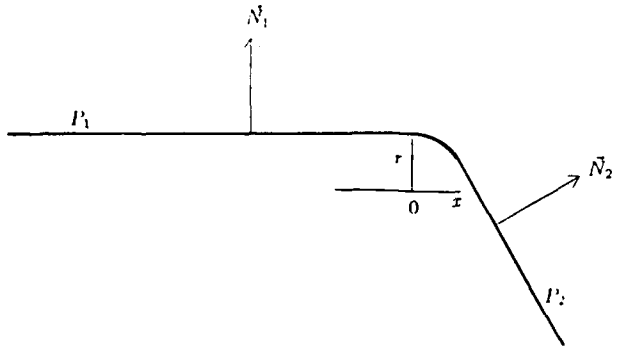


Figure 11. The Geometry of a Rounded Edge

The y axis is aligned in the direction of $\vec{N}_1$ and the origin is a distance r from P_1 such that the normal to the surface P_1 begins to turn away from $\vec{N}_1$ when x becomes positive. The model for the normal as it turns from $\vec{N}_1$ to $\vec{N}_2$ is

$$\dot{N} = \frac{\sqrt{r^2 - x^2}}{r}\vec{N}_1 + \frac{x}{r}\vec{N}_2 \quad for \;\; 0 \le x \le r|\vec{N}_1 \times \vec{N}_2| \quad (24)$$

In other words, the normal is assumed to turn through a curve of constant curvature $\frac{1}{r}$. Here the parameter r is used to specify the sharpness of the edge. Small values of r indicate sharp edges, while larger values of r indicate more rounded edges. Figure 12 shows the profile of a specularity across a sharp 90° edge which is similar to the profile predicted by the continuous normal variation of (24).

4.3. Predicting A

In the previous analysis we have assumed that over most configurations of viewer, source, and surface the adjusted geometrical attenuation factor A of (4) will have a small constant value across the specularity. For large angles of incidence, however, the character of A changes remarkably (Figure 13). In particular, for large

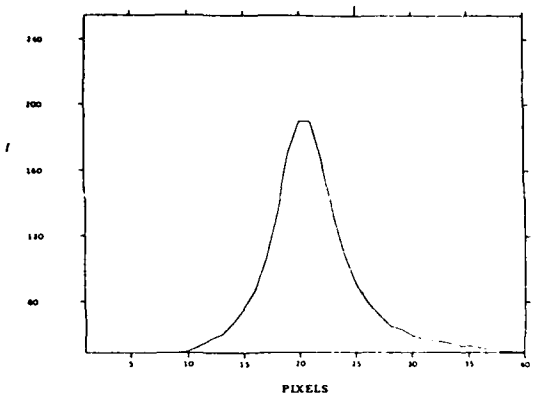


Figure 12. Specular Intensity Profile Across an Edge

angles of incidence (glancing incidence) we see that

1. A becomes large relative to its value for other incidence angles (Figure 13).
2. A causes a shift in the peak of the specular profile toward larger angles of incidence.
3. A causes the specular profile to be unsymmetric as a function of α.

It is not surprising that when these effects are present in an image, they are rather easy to detect. For this reason, it is useful to make qualitative predictions about A in applications where large angles of incidence are possible.

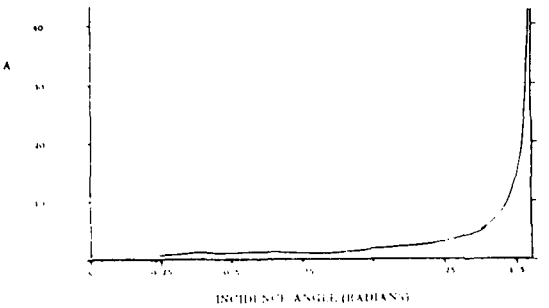


Figure 13. Plot of A as function of incidence angle

5. The Laboratory Setup

A laboratory arrangement has been set up to test the derived relationships (Figure 14). This section of the paper describes the laboratory setup. Section 6 examines factors which must be considered to successfully interpret real images. In Section 7, we describe an implemented system which has been used to infer local surface properties from specularities. Section 8 presents experimental results.

Figure 14. The Laboratory Setup

To insure accurate measurements, the experiments are conducted on a 4x6 foot optical table. High precision rotation and translation stages are used to position the objects being viewed. A halogen light source with a 5 mm wide filament is placed 20 feet from the object surface to approximate a point source. Monochromatic image data is obtained using a video camera and an image digitizer. A 210 mm lens is used with the video camera to obtain high resolution across the specularity. The resulting images are in the form of 256x256 arrays of pixels. Each pixel has eight bits of gray level resolution. A precise positioning device has been built to position the camera relative to the surface. Camera-object distances of at least 24 inches are enforced to insure that the assumed distant object condition is met. Using this setup, it is possible to obtain more than 10 pixels across a specular feature which is less than a centimeter wide on the surface. Metal cylinders and spheres of varying curvature are used to test the predicted relationships (Figure 15).

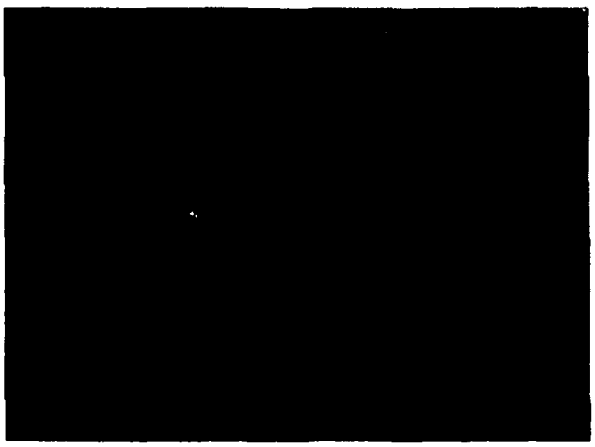


Figure 15. Some Experimental Specular Surfaces

6. Practical Considerations

This section examines factors which must be considered to enable a shape from specularity system to successfully interpret real images.

6.1. Gaussian Blur

Unfortunately, the formation of an image by an optical system introduces some amount of degradation. We can model this degradation as a convolution with a spatially invariant Gaussian point spread function [1]. The standard deviation of this blur is typically less than one pixel. For small specular features, taking into account the effects of this blur allows a more accurate determination of surface shape.

Our system uses a module called BLURINVERT to deblur the input specular image. For general 2-dimensional functions, inverting Gaussian blur is an unstable process. However, an explicit deblurring convolution kernel has been derived under certain assumptions in [15]. The 1-D continuous version of the kernel is given by

$$M_N(x) = \frac{e^{x^2}}{\sqrt{\pi}\, k!\, 2^k} \sum_{k=0}^{(N-1)/2} (-1)^k H_{2k}(x) \qquad (25)$$

where N is an odd integer denoting the order of the kernel and H_{2k} is the Hermite polynomial of degree 2k. Larger values of N give better deblurring filters (i.e. they recover exactly a larger space of blurred input functions), but are more costly to compute. The value of N that is chosen in applications depends on the intensity characteristics of the images that will be processed by the system. Using a 2-dimensional discrete version of (25), the BLURINVERT module allows our system to produce accurate shape descriptions from small specular features in images.

6.2. Quantization Effects

On a surface of high curvature, it is unlikely that we will measure the correct maximum specular intensity K' in (6). The problem is that for highly curved surfaces we are unable to shrink a pixel down to where the surface area it images is approximately planar. Even within the single pixel of maximum intensity, α is changing and cannot be considered constant. Hence the intensity value at the maximum pixel will be an average specular intensity over a small range of α and will not give the true K' of (6). This must be corrected for in applications. An artifact of this phenomenon is that measured K' seems to increase as surface curvature decreases. It

follows that if we wish to measure K' for a material, we should use a surface of small curvature, ideally a plane.

Since specularities are usually the brightest features in images, specular intensities are often too large to be represented in the number of bits per pixel allowed by the digitizing hardware or within the dynamic range of the camera. If this is the case, the specularity is truncated. Figure 16 shows I for a truncated specularity. The obvious way to deal with this situation is to avoid it. One avoidance technique is to take *multiple images* in which differing amounts of light are allowed to reach the camera. This can be achieved either by adjusting the lens aperture or by using filters. Another possible solution is to control the illumination to eliminate the possibility of truncation.

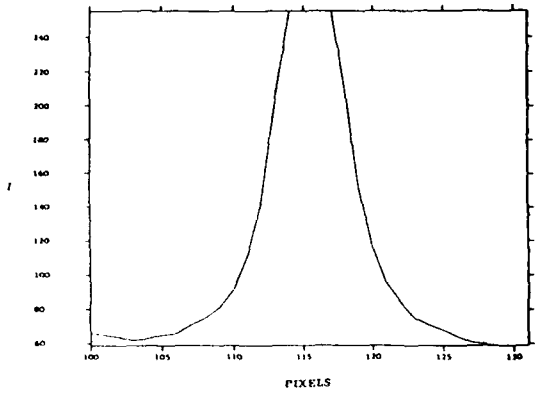


Figure 16. A Truncated Specularity

If inferences must be made from a single image, then it is arguably better to allow truncation to occur. In the case where input images have eight bits per pixel, intensities will range from 0 to 255. In many applications it is possible to weaken the incident illumination so that no truncation occurs. In doing this, however, we cause pixels on the I_S curve which previously had significant specular intensities (on the truncated specular feature) to have negligible specular intensities. The net effect of eliminating truncation is to decrease the width of the specular feature and make measurements more susceptible to small errors.

7. A Shape from Specularity System

A system has been implemented which computes local surface properties from images of specular surfaces [12]. The system currently stands alone, but will be used in the more general context of the SUCCESSOR vision system. The shape from specularity system is primarily designed to perform the computations described in Section 4. This section describes the implementation of the system.

7.1. Overview of System Structure

At a high level of abstraction, the problem is best solved in two steps. The first step is to deblur the input specular intensity image. The second step is to compute local surface properties from the image resulting from step 1.

The system designed to solve the problem preserves this two step structure. Figure 17 is a diagram of the modules in our system with arcs indicating module interactions. From this diagram we see that there is a *clean separation* between the deblurring task and the task of computing local surface properties. First the main program invokes a function called BLURINVERT to deblur the input image. After the deblurring task is completed, the function CURVATURES is called to compute local surface properties. The next two subsections give overviews of the BLURINVERT and CURVATURES functions.

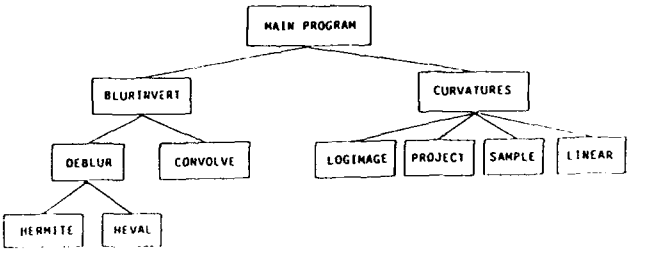


Figure 17. System Structure

7.2. Overview of BLURINVERT

The BLURINVERT function is used to deblur the Gaussian blurred input image. This is accomplished in two stages. First, the deblurring convolution kernel is generated by DEBLUR. Then the CONVOLVE function is called to perform the convolution of the blurred input image with the constructed deblurring kernel. Two functions are used by DEBLUR to manipulate the Hermite polynomials required to generate the deblurring filter. The function HERMITE uses a dynamic programming scheme to compute coefficients of all Hermite polynomials up to some specified degree. The function HEVAL is used to evaluate Hermite polynomials at fixed values of the polynomial parameter.

7.3. Overview of CURVATURES

Given a deblurred specular intensity image, the CURVATURES function computes the principal curvatures and directions of the surface at specular points. CURVATURES first uses the function LOGIMAGE to

transform the intensity image into the α angle image of Section 4. The CURVATURES function then systematically computes 1-D curvature at different directions in the tangent space to the surface. The function PROJECT is used to compute the metric transform between the tangent space to the surface and the image. This is necessary to insure that the system samples the angle image at equidistant angles on the surface. The function SAMPLE is used to sample the α angle image in a specified direction. Finally, the function LINEAR is used to compute the least squares curvature given the data generated by PROJECT and SAMPLE.

8. Experimental Results

The system described in Section 7 has consistently generated accurate surface descriptions from images of specular surfaces. In this section we give examples of our system's performance on real images of metal objects illuminated by a point source. Figures 18(a), 19(a), and 20(a) are images of circular cylinders of varying radii. Each cylinder is oriented such that its axis is perpendicular to the axis of the imaging device. Figure 21(a) is the image of a sphere. The actual statistics of the surfaces are given in Table 1. The dotted lines in the images indicate the direction of maximum curvature as determined by the system. Figures 18(b), 19(b), 20(b), and 21(b) are plots of intensity along the dotted lines in 18(a), 19(a), 20(a), and 21(a). Note that in Figure 21 the specularity is truncated, but we are still able to compute accurate surface statistics. Table 2 gives the second order surface statistics computed by the system. Error represents the percent of error in the computation of the largest curvature of the surface. The small errors can be attributed to quantization effects, noise introduced during the measurement process, and the various simplifications made to the specular model.

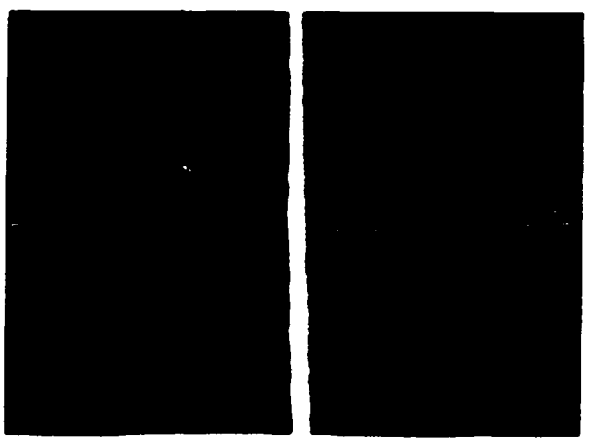


Figure 18(a)

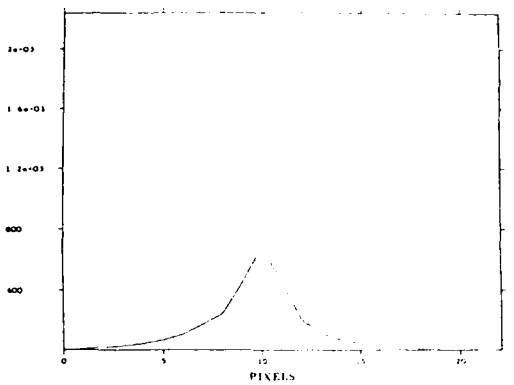


Figure 18(b)

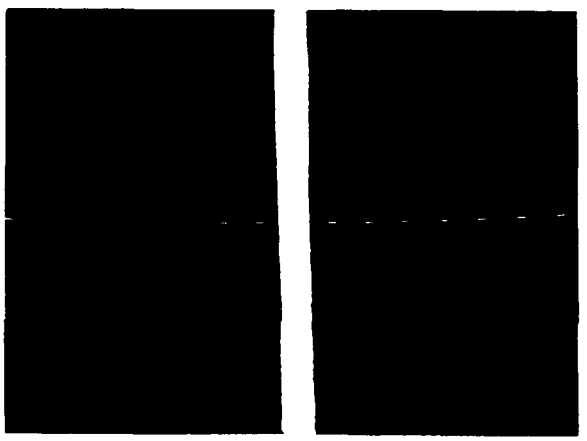


Figure 19(a)

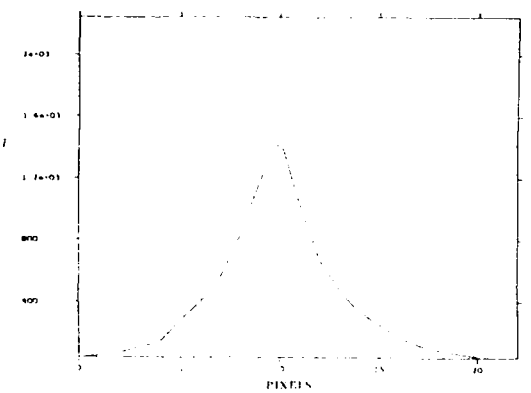
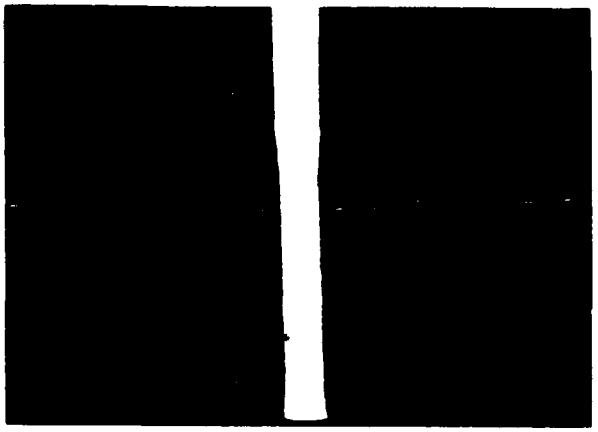


Figure 19(b)



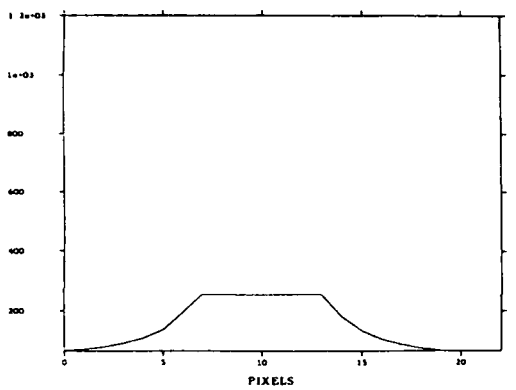
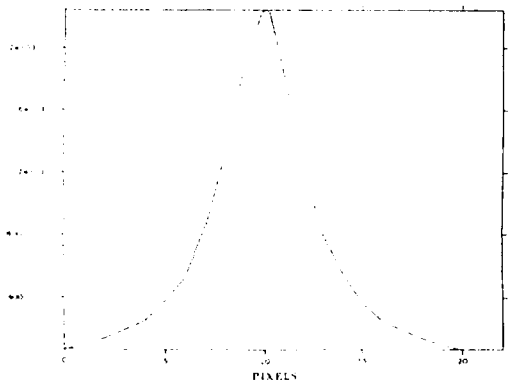


Figure 21(b)



Object	κ_1	θ_1	κ_2	θ_2
$cylinder_1$	0.286	0.0	0.0	1.571
$cylinder_2$	0.400	0.0	0.0	1.571
$cylinder_3$	1.333	0.0	0.0	1.571
$sphere_1$	0.500	—	0.500	—

Table 1. Actual Surface Statistics

Object	κ_1	θ_1	κ_2	θ_2	Error
$cylinder_1$	0.297	0.0	0.001	1.571	3.9%
$cylinder_2$	0.397	0.0	0.001	1.571	0.9%
$cylinder_3$	1.356	0.0	0.002	1.571	1.7%
$sphere_1$	0.514	0.0	0.534	1.571	2.8%

Table 2. Computed Surface Statistics

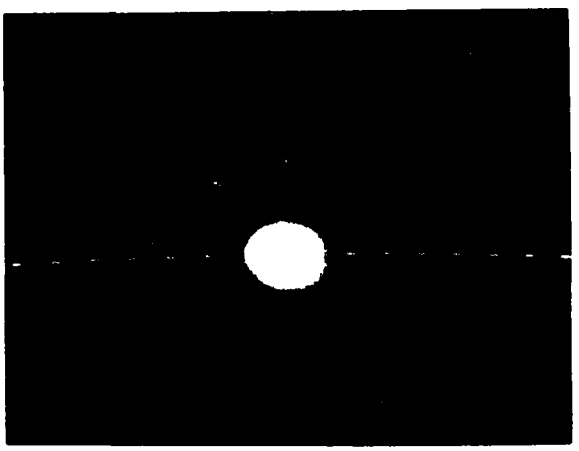


Figure 21(a)

9. Summary and Implications

Understanding specular reflection is important for any vision system which must interpret a world containing glossy objects. Using a model developed by optics researchers, we have shown that we can accurately predict the appearance of specular features in an image. In addition we have shown how to compute the local orientation and principal curvatures and principal directions of a specular surface by examining image intensities on a specularity. These statistics give a complete local characterization of the surface up to second order. Unlike previous work, our derivations have included the effects of surface roughness and microstructure on the appearance of specular features.

A system has been implemented which computes local surface properties from images of specular objects. A laboratory setup has been arranged which allows us to

capture images to test our system. The system has consistently produced accurate surface descriptions despite the fact that the high intensity and small spatial extent of specularities makes measurements difficult. Significant aspects of the implementation are discussed in Section 7. Examples of experimental results are given in Section 8.

The ability to predict intensity-based features such as specularities opens up interesting possibilities for model-based vision. Previous model-based vision systems have restricted their predictions to the structure of edges which will be observed for a given model. An ability to predict intensity-based features will significantly enhance the top-down capabilities of a model-based vision system. Clearly it is advantageous to be able to make stronger predictions about an image by using additional information about the imaging process. Another important advantage of predicting intensity-based features is that this prediction can provide strong guidance to low level intensity-based visual processes. By making predictions about the appearance of intensity patches in an image we can hope to further unify the goals of the low level and high level mechanisms of a model-based vision system.

More important than being able to predict the appearance of specularities from surface models is our system's ability to invert the process. We have shown how to infer local second order surface shape from specular images. This capability provides a vision system with strong generic information about a surface in a scene using strictly bottom-up processing. Inferring shape information from specularity is particularly important when viewing metal surfaces because other shape cues such as shading and texture are often not present. For other kinds of surfaces, shape information from specularity can be combined with shape information obtained using other cues to improve the 3-D surface descriptions generated by a vision system.

Acknowledgements

This work has been supported by an NSF graduate fellowship, AFOSR contract F49620-82-C-0092, and ARPA contract N000-39-84-C-0211. The authors would like to thank Professor Hesselink for generously providing laboratory space and equipment and Rami Rise for his work in designing the mechanical parts for the experiments. We would also like to thank Vishvjit Nalwa for providing useful comments on a draft of this paper.

References

[1] Andrews, H. and Hunt, B., "Digital Image Restoration," Prentice-Hall Inc., Englewood Cliffs, 1977.

[2] Blake A., "Specular Stereo," *Proceedings of IJCAI-9* (Los Angeles: August 1985), 973-976.

[3] Blinn, J., "Models of Light Reflection for Computer Synthesized Pictures," *Computer Graphics*, 11(2) (1977), 192-198.

[4] Brooks, R., "Symbolic Reasoning Among 3-D Models and 2-D Images," *Artificial Intelligence*, 17 (1981), 285-348.

[5] Bruss, A., "The Image Irradiance Equation: Its Solution and Application," MIT AI Memo 623, June 1981.

[6] Chelberg, D. and Lim, H. and Cowan, C., "ACRONYM Model-based Vision in the Intelligent Task Automation Project," *Proceedings of Image Understanding Workshop* (1984).

[7] Coleman, E.N. Jr. and Jain, R., "Obtaining 3-D Shape of Textured and Specular Surfaces Using Four-Source Photometry," *Computer Graphics and Image Processing*, 18 (1982), 309-328.

[8] Cook, R. and Torrance, K., "A Reflectance Model for Computer Graphics", *Computer Graphics*, 15(3) (1981), 307-316.

[9] Deift, P., and Sylvester, J., "Some Remarks on the Shape-from-Shading Problem in Computer Vision," *Journal of Mathematical Analysis and Applications*, Vol. 84, No. 1, pp.235-248, November, 1981.

[10] Foley, J. and Van Dam, A., *Fundamentals of Interactive Computer Graphics*, Addison-Wesley, 1982.

[11] Grimson, W.E.L., "Binocular Shading and Visual Surface Reconstruction," MIT AI Memo 697 (1982).

[12] Healey, G., "Solving the Inverse Specular Problem", Stanford Computer Science Department Programming Project, March 1986.

[13] Horn, B., "Obtaining Shape from Shading Information", in "The Psychology of Computer Vision" (P. Winston, Ed.), McGraw-Hill, New York, 1975.

[14] Horn, B., *Robot Vision*, MIT Press, 1986.

[15] Hummel, R. and Kimia, B. and Zucker, S., "Gaussian Blur and the Heat Equation: Forward and Inverse Solutions," *Proceedings of CVPR* (San Francisco: June, 1985).

[16] Ikeuchi, K., "Determining Surface Orientations of Specular Surfaces by Using the Photometric Stereo Method", *IEEE PAMI* 3(6) (1981), 661-669.

[17] Kreyszig, E., *Introduction to Differential Geometry and Riemannian Geometry*, University of Toronto Press, 1968.

[18] Phong, B., "Illumination for Computer Generated Pictures," *Communications of the ACM* 18 (1975), 311-317.

[19] Shafer, S., "Optical Phenomena in Computer Vision," Univ. of Rochester TR 135 (1984).

[20] Spivak, M., *Differential Geometry - Volume II*, Publish or Perish, Inc., 1979.

[21] Sparrow, E. and Cess, R., *Radiation Heat Transfer*, McGraw-Hill, New York, 1978.

[22] Takai, K. and Kimura, F. and Sata, T., "A Fast Visual Recognition System of Mechanical Parts by Use of Three Dimensional Model," source unknown. (First author is with CANON INC. in Tokyo.)

[23] Torrance, K. and Sparrow, E., "Theory for Off-Specular Reflection from Roughened Surfaces," *Journal of the Optical Society of America*, 57 (1967), 1105-1114.

[24] Woodham, R., "Photometric Stereo: A Reflectance Map Technique for Determining Surface Orientation from Image Intensity," Proc. SPIE, vol. 155 (1978).

FINDING OBJECT BOUNDARIES USING GUIDED GRADIENT ASCENT

Yvan Leclerc and Pascal Fua
Artificial Intelligence Center, SRI International
333 Ravenswood Avenue, Menlo Park, California 94025*

Abstract

We present a technique for finding object boundaries that combines both low- and high-level information. We integrate information along a candidate boundary and incrementally deform it, using a guided gradient ascent procedure, until a satisfactory curve is found. The strength of this technique lies in our ability to guide the search by applying forces to the boundary, guiding its change in shape and location. This technique overcomes inherent image ambiguities, thereby finding boundaries that could not otherwise be found.

1 Introduction

In real-world images, physical event discontinuities cannot be detected solely by their local statistical signature in the image because of the presence of noise and various photometric anomalies. All methods for finding discontinuities that use purely local statistical criteria are therefore bound to make mistakes and miss some of the weaker parts of edges.

To supplement the weak and noisy local information, we must integrate information from the many points that lie on an edge without introducing irrelevant and often misleading information from other points. We propose to do this by integrating information along a candidate curve and incrementally deforming it, using a guided gradient ascent procedure, until a satisfactory curve is found. The strength of our procedure lies in the ability to guide the search, perhaps on the basis of semantic information, by applying forces to the curve to guide its change in both shape and location.

In the next section, we describe our model of an ideal step-edge. We then describe the guided gradient ascent procedure, using the search for a straight edge as an example. Finally, we present applications of the procedure to automated cartography and object location.

2 Our Model of An Ideal Step-Edge

We define an ideal step-edge to be a simple one-dimensional curve C for which the following properties hold everywhere along the curve [Haralick, 1984]:

1. The magnitude of the directional first-derivative of image intensity normal to the curve is a local maximum in that direction; i.e., the directional second-derivative of image intensity normal to the curve is zero.
2. The magnitude of the directional first-derivative of image intensity tangential to the curve is zero.

Ideally, we wish to define a functional $\mathcal{F}(C; I)$ such that it has local maxima with respect to infinitesimal deformations of C when and only when these properties hold. Given such a functional, we can then formulate the problem of finding an ideal edge as one of maximizing $\mathcal{F}$ using gradient ascent methods.

We can define such a functional, but it is expensive to compute because it involves calculating the directional first-derivative of image intensity at every point along the curve as a function of its tangent. Instead, we define a functional that is much less expensive to compute, but that depends on conditions that are only necessary for a curve to be an ideal step-edge. Thus, once a candidate curve has been found by maximizing this simpler functional, we must test it using the more expensive and reliable criteria specified above.

Two necessary, but not sufficient, conditions for a curve C to be an ideal step-edge are:

1. the average magnitude of image intensity gradient ($|\nabla I(x,y)|$) along C must be a local maximum with respect to infinitesimal deformations of C, and
2. the average variance of the magnitude of image intensity gradient along C must be a local minimum with respect to infinitesimal deformations of C;

with the corresponding functionals:

$$\mathcal{F}_m(C; I) = \int_C |\nabla I(x,y)|\, ds / |C|$$

$$\mathcal{F}_v(C; I) = \int_C |\nabla I(x,y) - \mu|^2\, ds / |C|,$$

$$\mu = \mathcal{F}_m(C; I), \qquad |C| = \text{length of } C.$$

In principle, one should be able to use either one or the other functional alone, since they correspond to necessary, but not sufficient, conditions. In practice, however, using either one alone produces too many local extrema that do not satisfy the full conditions. An intuitive explanation for this is that maximizing $\mathcal{F}_m(C; I)$ alone can produce curves that span several disjoint edges whose average magnitude might be quite high, whereas minimizing $\mathcal{F}_v(C; I)$ alone can produce arbitrary curves over entirely uniform areas of the image. For these reasons, we have chosen to maximize the combination

$$\mathcal{F}(C; I) = \frac{\mathcal{F}_m(C; I)}{[\mathcal{F}_v(C; I)]^\alpha} \qquad 0 < \alpha < 1,$$

which almost always produces satisfactory curves when started sufficiently near an edge. Choosing a large α tends to force intermediate curves to be approximately parallel to the final curve, providing stability in the search. However, α should be chosen to be small enough that somewhat nonideal edges, which are prevalent in real images, can still be found. In practice, $\alpha = 0.2$ produces satisfactory results in most situations.

The great advantage of this functional over the ideal one is that it can be computed very cheaply because $|\nabla I(x,y)|$ need be computed only once for every image point, independently of the curve. Thus, we can approximate the integrals by summing over a precomputed gradient-magnitude image such as that produced by a Sobel operator. Furthermore, in the final stages of the procedure described in the next section, we can interpolate the results of the Sobel operator to produce subpixel estimates of the gradient-magnitude, and hence subpixel estimates of the edge.

3 The Guided Gradient Ascent Procedure

We are interested in finding curves that maximize $\mathcal{F}(C;I)$ given an initial estimate of the curve's location and shape [Witkin *et al.*, 1986]. For a curve defined by a number of geometric parameters, we can do this by using gradient ascent given initial estimates of these parameters.

The unguided gradient ascent algorithm iteratively takes steps of a given size in the direction of the gradient. As the iterations proceed, the step-size is reduced whenever $\mathcal{F}$ decreases *and* when certain quality criteria, such as $\mathcal{F}_m$ being greater than some minimal value, are satisfied. The latter criterion is important to avoid stopping at weak local maxima, which occur relatively frequently because of noise. The procedure is stopped when the step-size becomes smaller than some minimal value, typically corresponding to some fraction of a pixel for position and a few degrees for orientation.

For example, consider the case of a straight line-segment of fixed length. Figure 1(a) shows the image that we will be using for all of our examples, and Figure 1(b) shows the corresponding gradient-magnitude image.

Figure 2(a) shows such a line-segment placed near a straight edge in an image. Figure 2(b) shows an intermediate step as the gradient ascent procedure iterates, with the final result in Figure 2(c).

Of course, this procedure crucially depends on being relatively near the correct final result. For example, if we were to place the initial line-segment in a uniform area, it would oscillate and might never stop.

The strength of our approach, however, lies in our ability to guide the procedure when some *a priori* semantic or geometrical information is known. This is done by applying forces to the curve by multiplying the base functional by forcing functionals that depend on the parameters of the curve, but not on the image.

To continue our previous example using forcing functionals, we want to find an edge parallel to the one in Figure 2(c). We can guide the procedure by applying a force to each of the end-points in the direction perpendicular to the original line (multiplying $\mathcal{F}$ by the length of the vector that is perpendicular to the original line passing through the end-point). By starting with a line-segment close to the one in Figure 2(c), and applying this force, the system found the parallel edge illustrated in Figure 3. Note that the gradient-magnitude of this edge is extremely weak, and would therefore have been missed entirely by local techniques.

4 Applications of the Approach

We have implemented several more sophisticated applications of the guided gradient ascent approach, of which only two can be briefly discussed here.

The first application is to automated cartography. In an aerial image, the height of a building can be estimated by using the known sun direction, the position of one of its corners, and its corresponding shadow corner. In order to build an interactive cartographic system that will perform this task automatically, we want to allow the user to point near a corner and have the system find its exact location and orientation.

The procedure we have implemented starts by having the user point inside a building or shadow near a corner. A rough estimate of the directions of the edges is computed by finding the two main peaks in the histogram of Sobel directions in a window around this point. These two directions are used to split the window into four quadrants, corresponding to the four possible orientations of the corner. If the corner is within the window, three of the quadrants overlap its edges and only one is within the object or the shadow. A simple area-based measure determines which quadrant is most uniform, and hence entirely within the object or shadow. Figure 4(a) shows the initial guess computed for two user-supplied points.

The initial guess is fed to the optimization algorithm, with forces applied to the candidate corner such that its edges are pushed outward. The edges are coupled by a force proportional to the angle of the corner. This prevents one edge from stopping in a local minimum when the other has not yet stabilized. Figure 4(b) shows the final result, in which the corners have been accurately found although the initial points were quite distant from the actual corners.

The second application, described in detail in this proceedings [Fua and Hanson, 1987], is to integrate this approach in a fully automated system for the detection and location of cultural objects in aerial images. Briefly, the system hypothesizes the shape of objects by combining an Ohlander-style segmentation with geometric models and semantic information about the expected class of objects. The guided gradient ascent is then used to improve and verify these predictions. The technique has allowed the system to discover boundaries that were entirely missed by both the segmentation and local edge operators.

5 Conclusions

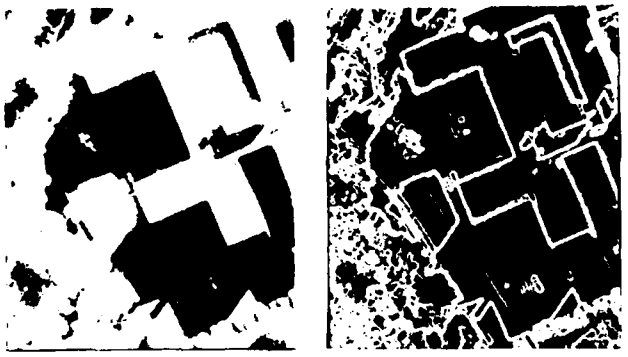
We have presented a technique for finding object boundaries given rough initial estimates of their shape and position. The power of this technique lies in its ability to combine low- and high-level information. In many scenes, the ambiguities of the data are such that shapes cannot be extracted reliably by the sole use of low-level cues. Our technique allows us to supplement this weak information with higher-level knowledge, such as the expected shape of the boundaries, which is used to determine the shape of the operator and the guiding forces. This additional information allows the system to overcome inherent image ambiguities and find shapes that could not otherwise be found.

References

P. Fua and A. Hanson, "Using Generic Geometric Models for Intelligent Shape Extraction," this proceedings, 1987.

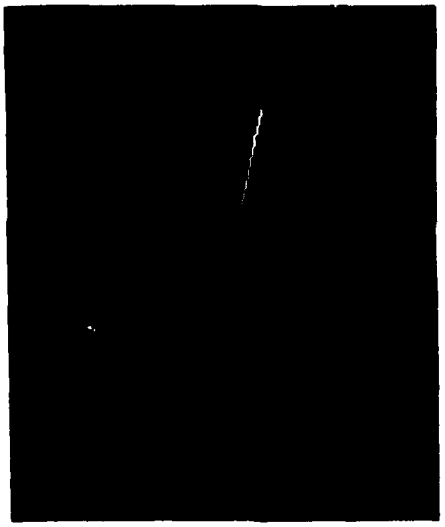
R.M. Haralick, "Digital Step Edges from Zero Crossings of Second Directional Derivatives," *IEEE Trans. PAMI*, Vol. 1, pp. 58-68, 1984.

A. Witkin, M. Kass, D. Terzopoulos, and A. Barr, "Modelling with Dynamic Constraints: Linking Graphics to Perception," *Proc. of the Rank Prize Funds International Symposium on Images and Understanding*, Royal Society, London, England, September, 1986.

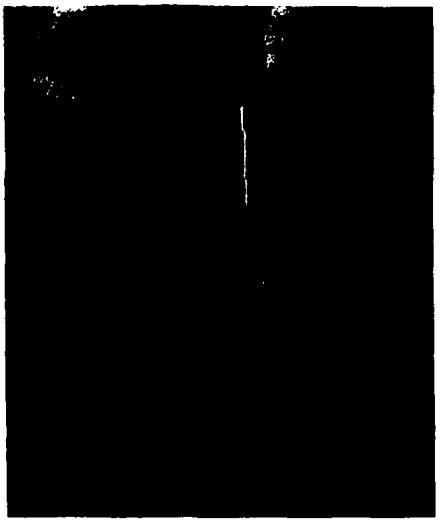


(a) (b)

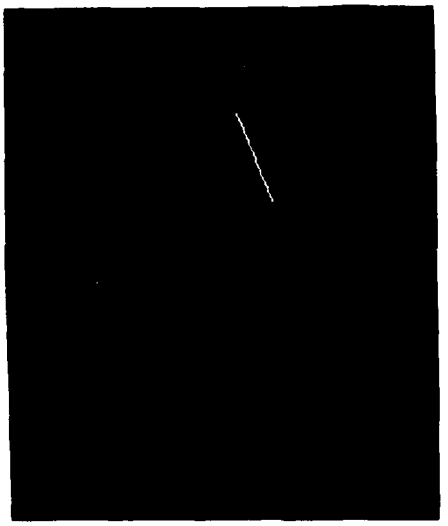
Figure 1: (a) The original image (b) The gradient image



(a)



(b)

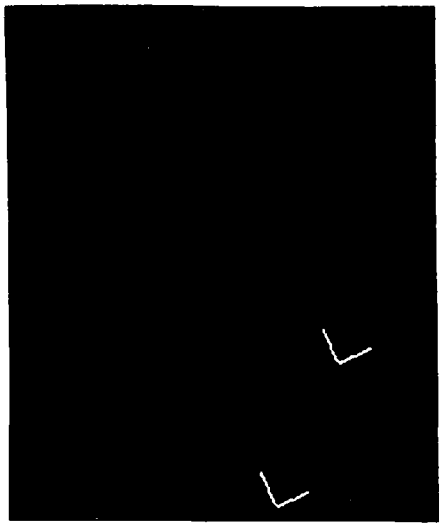


(c)

Figure 2: (a) The initial guess for the edge
(b) An intermediate step
(c) The final location of the edge

Figure 3: A weak parallel edge

(a)



(b)

Figure 4: (a) The initial guess for the corners
(b) The final location

The work reported here was partially supported by the Defense Advanced Research Projects Agency under contracts DACA76-85-C-0004 and MDA903-86-C-0084.

DETECTING BLOBS AS TEXTONS IN NATURAL IMAGES

Harry Voorhees and Tomaso Poggio

Massachusetts Institute of Technology
Artificial Intelligence Laboratory

Abstract. Texture provides one cue for identifying the physical cause of an intensity edge, such as occlusion, shadow, surface orientation or reflectance change. Marr, Julesz, and others have proposed that texture is represented by small lines or blobs, called "textons" by Julesz [1983], together with their attributes, such as orientation, elongation, and intensity. Psychophysical studies suggest that texture boundaries are perceived where distributions of attributes over neighborhoods of textons differ significantly.

However, these studies, which deal with synthetic images, neglect to consider an important question: How can these textons be extracted from images of natural scenes? This paper proposes an answer to this question by proposing an algorithm for detecting blobs in natural images. As part of the blob detection algorithm, methods for estimating image noise are presented, which are applicable to edge detection as well.

1 Introduction

A crucial step in the visual process is the identification of physical discontinuties in the scene—sharp changes in surface depth and orientation, reflectance, and illumination. All of these phenomena can result in intensity edges, so other processes besides edge detection are needed to identify these events. Texture, along with color and motion, provides a cue. As discussed by Riley [1981], drastic texture changes can only be explained by occlusion, while projection-type changes can be due to a surface orientation discontinuity as well. It is with this goal in mind—the interpretation of physical discontinuities—that we investigate the visual processing of texture information.

In his theory of the primal sketch, Marr [1976] proposed that texture is represented in the raw primal sketch—a symbolic description of localized intensity changes in the image, including edges, compact blobs, and linear blobs called bars. Attributes of these tokens–their size, orientation, contrast, and termination points–are computed, along with their position. Texture boundary detection and the extraction of form proceeds by comparing first order statistics of distributions of these attributes—size, orientation, contrast, and density—over nearby neighborhoods.

Since the primal sketch theory was proposed, a number of psychophysicists have studied human texture perception. Julesz (who had previously held the opposite view) demonstrated that first order local statistics of blobs better account for perceived texture differences than second order global statistics of pixels [Julesz, 1983]. He concluded that blobs called "textons"—together with their geometric and intensity attributes, termination points, and crossings—form the primitives on which texture perception is based. Many psychophysical studies of texture perception have assumed the existence of texton detectors, employing synthetic images of symbols such as line segments.

However, these psychophysical studies, which are based only on synthetic images, neglect to consider an important issue: How can such textons be reliably extracted from images of natural scenes? Here we attempt to to answer this question by describing an algorithm we have implemented which detects small blobs in natural images.

2 Noise estimation

When dealing with images of natural scenes, it is necessary to estimate the amount of noise in an image. In edge or blob detection, the noise estimate determines a threshold above which an intensity change is considered "significant." In this section, we show how noise can be estimated from a histogram of the image convolved with a gaussian or laplacian of gaussian filter, depending on which is used for edge or blob detection.

2.1 Gaussian filtering

If one assumes that the smoothed image $f(x,y) = I * G$ consists of white gaussian noise,

$$\Pr_f(z) = G(\sigma; z) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}(\frac{z}{\sigma})^2} \qquad (1)$$

then, using the probablility law for a function of random variables, the partial derivatives f_x and f_y have p.d.f.'s

$$\Pr_{f_x}(z) = \Pr_{f_y}(z) = \int_{-\infty}^{\infty} \Pr_f(z) \Pr_f(z-s) ds = G(\varsigma; z), \qquad (2)$$

where $\varsigma = \sqrt{2}\sigma$. Again using the probability law, the

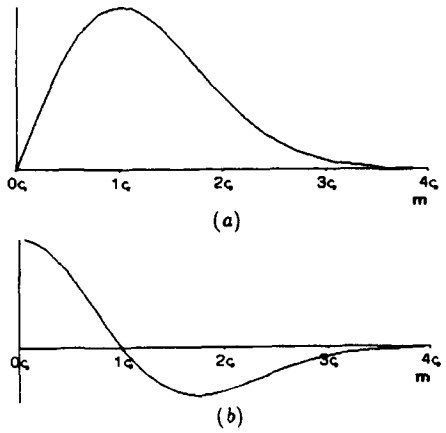


Figure 1: The Rayleigh distribution

(a) Rayleigh distribution and (b) its first derivative.

p.d.f. of the magnitude of the gradient $\|\nabla f\| = \sqrt{f_x^2 + f_y^2}$ is

$$\Pr_{\|\nabla f\|}(m) = z \int_0^{2\pi} \Pr_{f_x}(m\cos\theta) \Pr_{f_y}(m\sin\theta) d\theta$$
$$= mG(\varsigma; m), \quad \text{for } m > 0. \tag{3}$$

The magnitude of gradient distribution is a Rayleigh distribution, graphed in Figure 1, with a maximum value at $m = \varsigma$. The addition of real edges, whose gradients are stronger than those of the noise distribution, affect mainly the tail of the distribution, and do not significantly affect the location of the peak. Bracho and Sanderson [1985], who applied this idea to region growing, noted that the assumption holds as long as a large portion of the image consists of roughly uniform intensity which satisfies the white noise model. Often the background of the image is such a region. In this case, the noise can be estimated by constructing a histogram $h(m)$ of $\|\nabla f\|$, and simply measuring the location of the peak as ς. Figure 2 shows an example for which this simple method works well.

While this strategy works well for images containing some uniform regions, it can fail when the image contains many strong texture edges, as shown in Figure 3(a). In this case, the many edges of strong gradient obliterate the the local maximum of $h(m)$ due to the noise distribution. It is therefore preferable to fit the Rayleigh distribution to the steep, rising portion of the histogram.

The fitting can be done easily by using the first derivative of the smoothed histogram, $h'(m)$. Either the location where h' crosses zero or where it "would have crossed" zero, whichever quantity is less, estimates ς. The latter position is computed by linearly extrapolating the downward sloping portion of h', as shown in Figure 3(c). The former case, of course, corresponds to the first local maximum of $h(m)$, which is used as the estimate for images satsifying the uniform region assumption.

Once ς is estimated, one can choose a threshold to exclude most edges due to noise. To remove noise with a confidence of c, requires a threshold τ such that

$$c = \frac{\int_0^\tau zG(\varsigma; z)dz}{\int_0^\infty zG(\varsigma; z)dz}, \tag{4}$$

yielding $\tau/\varsigma = \sqrt{-2\ln c}$. Hence, removing noise with a confidence of 99% requires $\tau/\varsigma \approx 3$.

Figures 3(f) and 2(c) demonstrate that this improved method works for images consisting entirely of highly textured images, as well as for images containing untextured areas. In these examples, we used the edge detector of Canny [1983], thresholding the edges with hysteresis, using $\tau_1/\varsigma = 2$ and $\tau_2 = 2\tau_1$. In many applications, it is desirable to remove weak "real" edges as well, in which case higher threshold factors are used. The examples also show that simply choosing a fixed percentile of the histogram does not provide a robust estimate of noise.

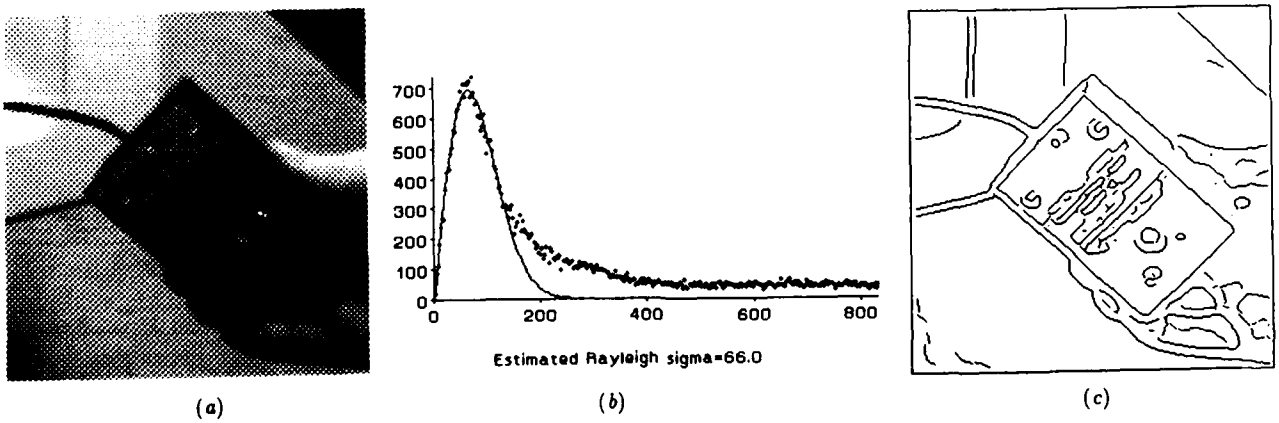


Figure 2: Simple Noise Estimation Example

(a) Image of mouse, containing smooth intensity surfaces. (b) Magnitude of gradient histogram with Rayleigh distribution fit to peak at $m = \varsigma$. (c) Edges detected using hysteresis with $\tau_1 = 2\varsigma$, $\tau_2 = 4\varsigma$.

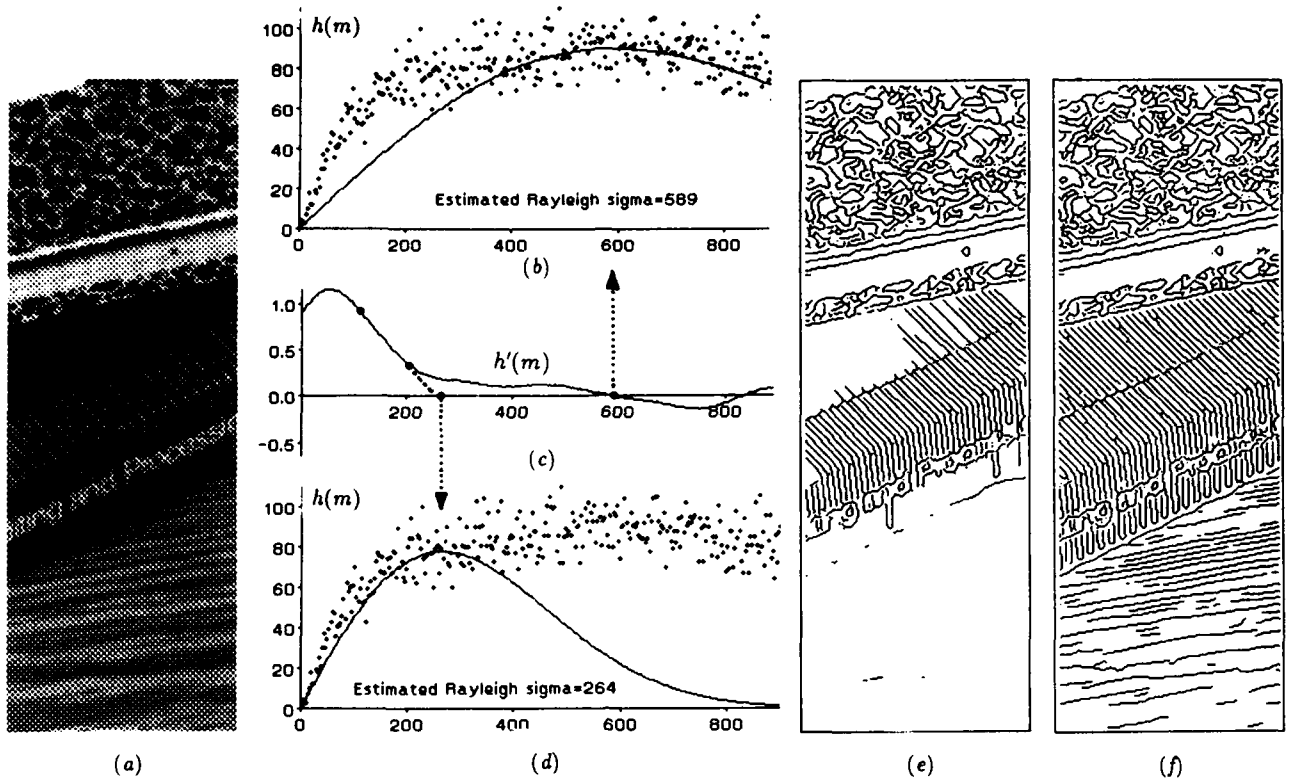


Figure 3: More Complicated Noise Estimation Example

Since this image (a) contains no smooth intensity surfaces, fitting the Rayleigh distribution to the global maximum of the gradient histogram, $h(m)$ (b) results in too high a edge threshold (e). A better fit is obtained (d) by extrapolating the first derivative of the smoothed histogram (c), resulting in a better edge threshold (f). Edge images (e) and (f) were thresholded with hysteresis with $\tau_1 = 2\varsigma$, $\tau_2 = 4\varsigma$.

2.2 Laplacian of Gaussian filtering and blob detection

For laplacian of gaussian edge or blob detection, one is interested in choosing a non-zero threshold τ, to exclude most responses due to noise. If, as before, we assume the smoothed image $f = I * G$ consists of white gaussian noise with standard deviation σ, then the p.d.f of $\nabla^2 f \equiv f_{xx} + f_{yy}$ is $\Pr_{\nabla^2 f}(z) = G(\varsigma; z)$, where $\varsigma = 4\sigma$. In this case the histogram of $\nabla^2 f$ is a gaussian distribution. Again, real edges mainly affect the tails of the distribution, so the gaussian distribution is fit to only the central portion of the histogram.

The relationship between threshold τ and confidence level c is given by the standard normal distribution. Avoiding responses due to noise with $c = 99\%$, for example, requires $\tau/\varsigma \approx 2.3$.

3 Blob detection

For blob detection it is natural to use the sign bits of $I * \nabla^2 G$. In this sense blobs may be regarded as duals of edges, which can be detected using the zero crossings of $I * \nabla^2 G$. Instead of thresholding at zero, however, a small threshold is used which removes some connections between blobs due to noise. The threshold is positive for dark blobs (and negative for light blobs), and its magnitude is proportional to the noise estimate. In the examples here, we identify dark blobs where $I * \nabla^2 G > \ell_0$, where $\ell_0 \approx \varsigma$.

For the purpose of relating texture to the properties of physical surfaces, it is desirable to discount the effects of illumination. In the simplest model of image formation, brightness is roughly a multiple of reflectance and incident illumination, $b(x,y) = r(x,y)i(x,y)$ [Horn, 1986]. The presence of a shadow decreases the brightness difference between a texton and its background and therefore decreases the response of $I * \nabla^2 G$ over the shadowed region. Hence, if pixel values are linear with brightness, shadows can cause texture density or contrast boundaries, as

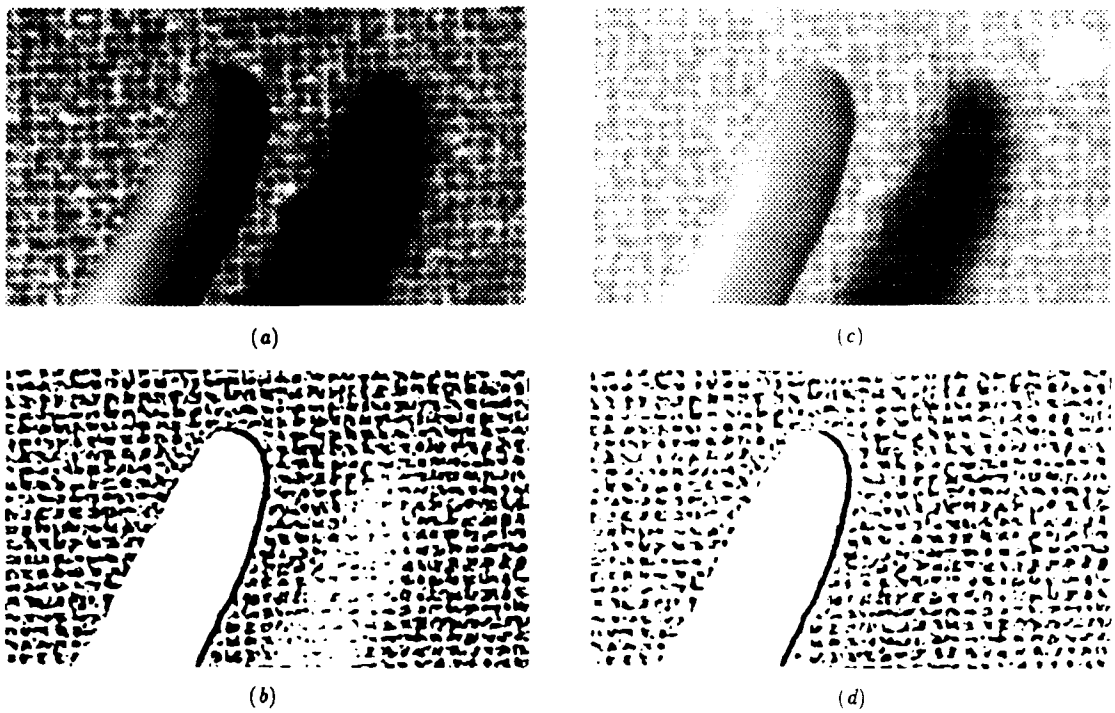


Figure 4: Discounting Effects of Illumination

(*a*) Image I of finger casting a shadow on burlap. Intensity values are approximately linear with brightness. (*b*) The shadow reduces the density of blobs if $I * \nabla^2 G$ is used. (*c*) Logarithm of image, $\log I$. (*d*) Density of blobs remains the same if $(\log I) * \nabla^2 G$ is used.

shown in Figure 4(a, b). This problem can be partially avoided by using image values proportional to the logarithm of brightness. Then differences in the image values are proportional to ratios of brightness values, which are illumination-independent:

$$\log b_1 - \log b_2 = \log \frac{b_1}{b_2} = \log \frac{r_1}{r_2}, \qquad (5)$$

where b_1, b_2, r_1, and r_2 are the brightness and reflectance values of a texton and its surround. As a result, shadows do not affect the density and contrast of blobs, as shown in Figure 4(c, d). Implementing such a scheme, of course, requires that cameras be calibrated. Most cameras have neither purely linear nor logarithmic response, suggesting the use of a look-up table for transforming their output to $I = \log b$.

3.1 Blob segmentation

Although some noise has been removed, one cannot simply treat connected components of the thresholded $\nabla^2 I * G$ array as textons, since these regions can span large distances in the image. In order to compute meaningful attributes of blobs, such regions must be segmented into localized blobs and bars. Thin joins between otherwise compact bars must be eliminated. Also, elongated regions must be separated at bends and intersections into roughly linear bars. Simple procedures for performing these two types of segmentation are described next.

3.1.1 Compact blob segmentation

Often, for a particular level, two nearby extrema of $I * \nabla^2 G$ may be joined into a single blob, containing a thin neck. These connections are sensitive to the level chosen and may result in connected regions which span large portions of the image. Such regions are segmented using multiple levels. Whenever a higher level would result in a region being split into two or more regions, we use the new regions provided that they are relatively stable. The stability criterion (similar to the stability criterion for fingerprints proposed by Witkin [1983]) requires that the new regions exist over at least as large a range of levels than the underlying one. The splitting operation is applied recursively at several levels. Finally, the new, smaller regions are numbered and then grown back over the original regions in order to segment them. The process is illustrated in Figure 5.

3.1.2 Bar segmentation

Elongated regions can be segmented at junctions and bends by using a refinement of the medial axis transform. The medial axis, obtained by thinning the region down to its skeleton, has been criticized for being sensitive to slight perturbations in the region contour (see Figure 6). However, a stable representation can be obtained by keeping only those segments of the skeleton which represent elongated sections of the region.

The pruning is accomplished as follows: Straight line segments are fit to the skeleton, with knot points placed at any intersections as well. The average distance along each corresponding skeleton segment to the perimeter of the region is computed, and only segments whose lengths are longer than α times their radii are retained (we use $\alpha = 3$, which corresponds to an minimum aspect ratio of $\alpha/2 = 3/2$). At this stage, a maximum aspect ratio can be imposed on textons, if desired, by splitting very long segments. The algorithm removes insignificant skeleton segments, retaining only those which represent elongated, bar-like parts of the region. The remaining segments, numbered, are then grown back over the original region, segmenting it into approximately linear segments. Compact regions, which have no significant skeleton segments, are not segmented. Figure 7 shows a simple example.

The blob detection and segmentation operations are illustrated, step-by-step, on a natural image in Figure 8.

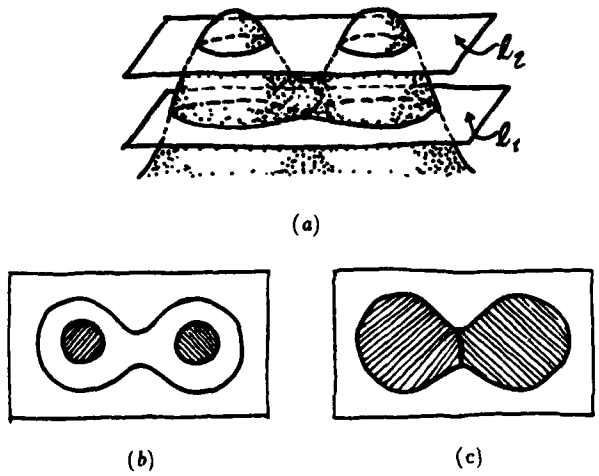


Figure 5: Segmenting compact bars using multiple levels

(*a*) The surface $I * \nabla^2 G$ sliced at levels ℓ_1 and ℓ_2. (*b*) The blob at ℓ_1 is split into two blobs at ℓ_2. (*c*) The blobs at ℓ_2 are labelled and grown over the original blob to segment it.

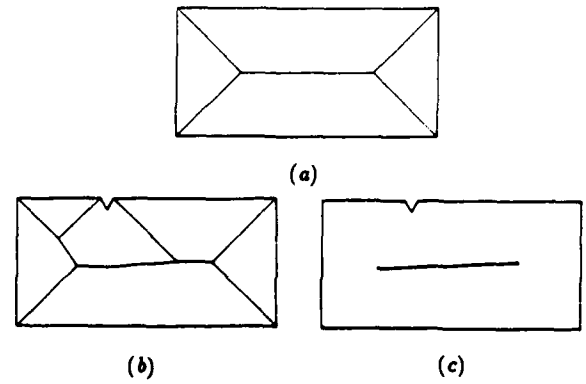


Figure 6: A More Robust Medial Axis Transform

The medial axis transform of a rectangle (*a*) is undesirably sensitive to slight changes in the bounding countour (*b*) [From Marr, 1982, p. 304]. Fitting straight line segments to the skeleton and removing those which are short relative to their distance from the perimeter yields a more robust representation (*c*).

3.2 Attribute computation and removal of spurious blobs

At this point the blob regions have been segmented into localized blobs, whose attributes can be computed. The length, width, and orientation of each blob is computed from its second moments (see, for example, [Horn, 1986]).

In addition to geometric attributes, the average contrast of each blob is computed along its perimeter using the quantity $c(s) = \nabla I * G \cdot \vec{n}(s)$, where $\vec{n}(s)$ is the normal along the blob's perimeter. Assuming a step edge of amplitude

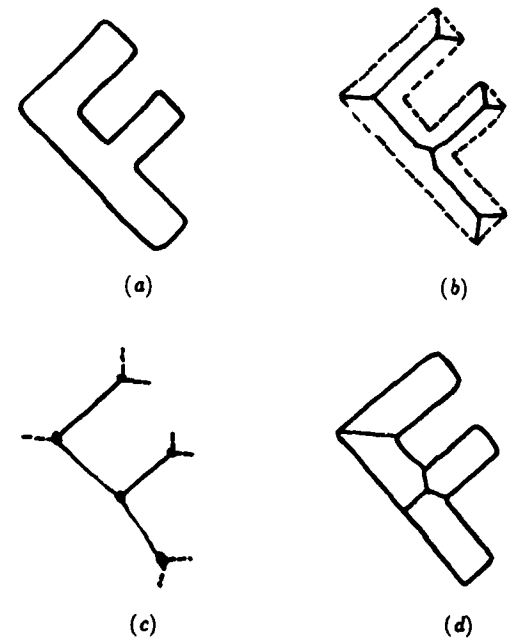


Figure 7: Segmenting Bars at Junctions and Intersections

(*a*) A bar-like figure to be segmented. (*b*) Skeleton (medial axis transform) obtained by thinning. (*c*) Straight line skeleton with segments having small aspect ratios removed. (*d*) Segmentation obtained by growing labelled segments back over original figure.

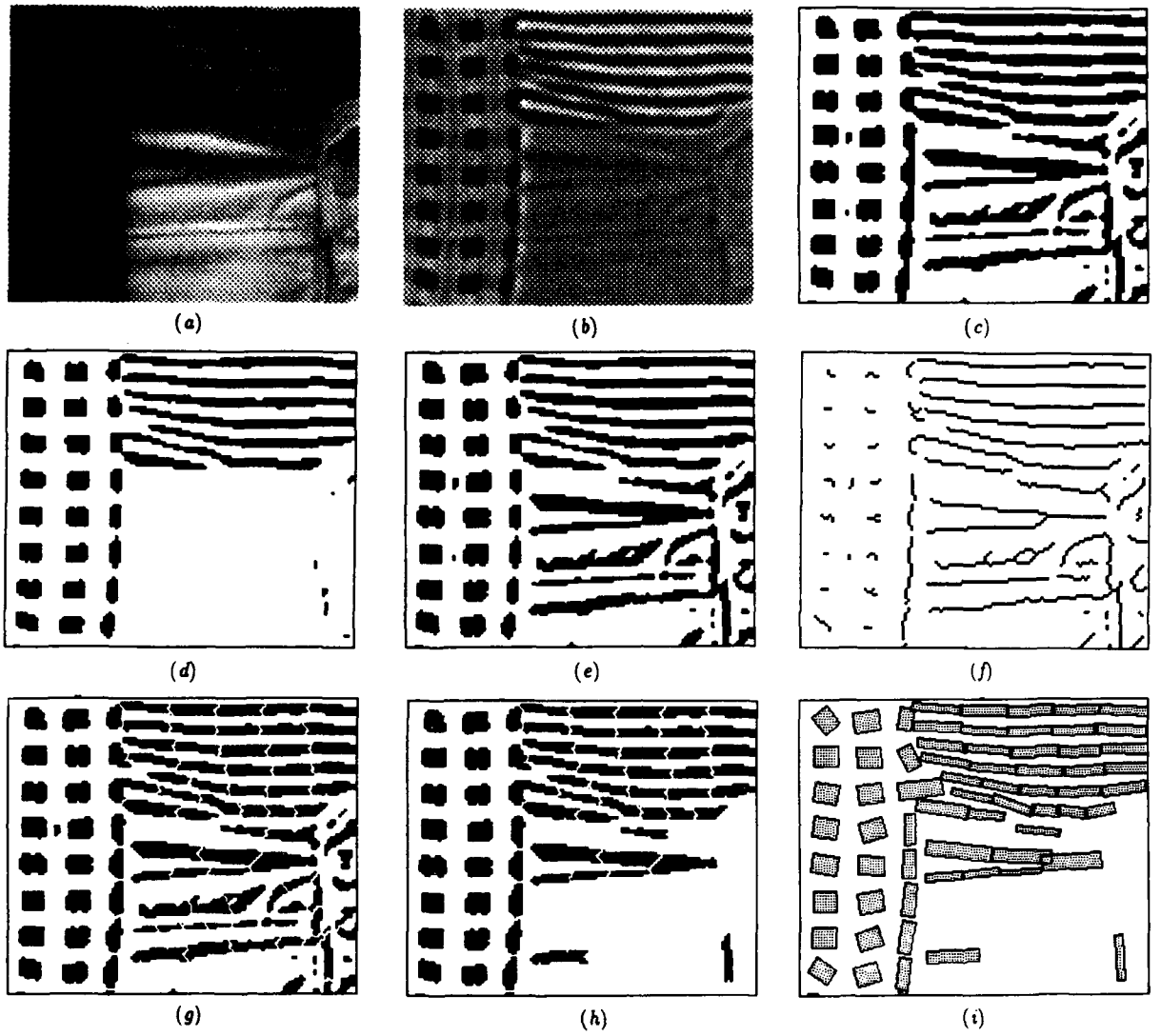


Figure 8: Steps of blob detection and segmentation

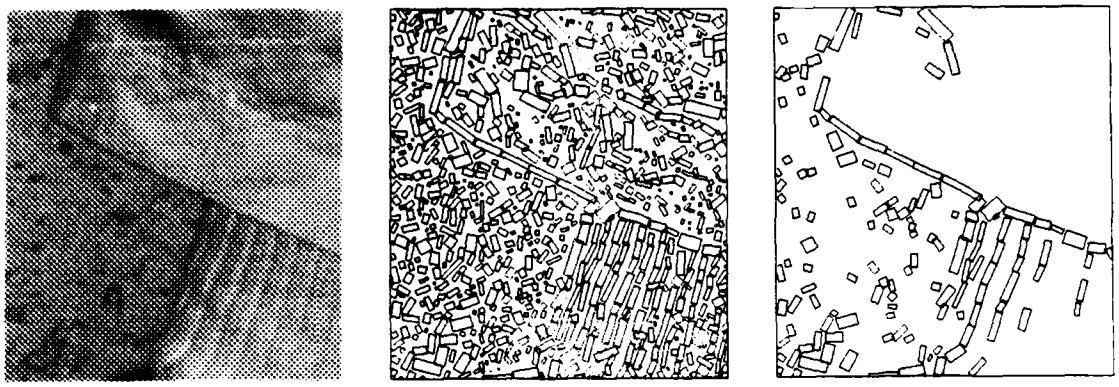
(*a*) Image I, a close-up of mismatched clothing. (*b*) $I * \nabla^2 G$. (*c*) $I * \nabla^2 G > \ell_1$. (*d*) $I * \nabla^2 G > \ell_2$, $\ell_2 > \ell_1$. (*e*) Segmentation obtained by growing (d) over (c). (*f*) Significant skeleton segments of (e). (*g*) Segmentation obtained by growing (f) over (e). (*h*) Result of removing spurious blobs from (g). (*i*) Rectangles showing geometric attributes of blobs in (h).

a, the contrast at a point on the perimeter $c(s) = a/2\pi\sigma$, where $G(\sigma)$ is the gaussian smoothing filter.

Only blobs having over half their perimeter above the noise threshold (computed as in Section 2.1) are maintained. This removes remaining blobs due to noise as well as any spurious blobs caused by strong edges along only one side of them. Figures 9 and 10 show more examples.

3.3 Analysis of blob detection algorithm

The proposed blob detection scheme is not claimed to be biologically plausible. A more biologically plausible algorithm is offered by Mahoney [1987], who uses elliptical masks at different resolutions and orientations to detect blobs in a binary array. However, the algorithm proposed here employs local operations and is quite efficient. As currently implemented on a Symbolics lisp machine, the entire blob detection process takes about 6 minutes on the 160×454 pixel books image shown in Figure 10(a). Of this time, about 2 minutes are used for convolution, differentiation, and noise estimation; 3 minutes for blob segmentation (using 5 levels); and 1 minute for attribute computation and thresholding. On the Connection Machine, we expect the algorithm to take a few seconds.



(a) (b) (c)

Figure 9: Blobs due to surface roughness

(a) An image of a pant leg and sock against a pitted ceiling tile. Color does not provide a cue for locating discontinuities in the scene, since all surfaces are white. (b) All blobs found. (c) Strong blobs found at scale $\sigma = 2$ pixels.

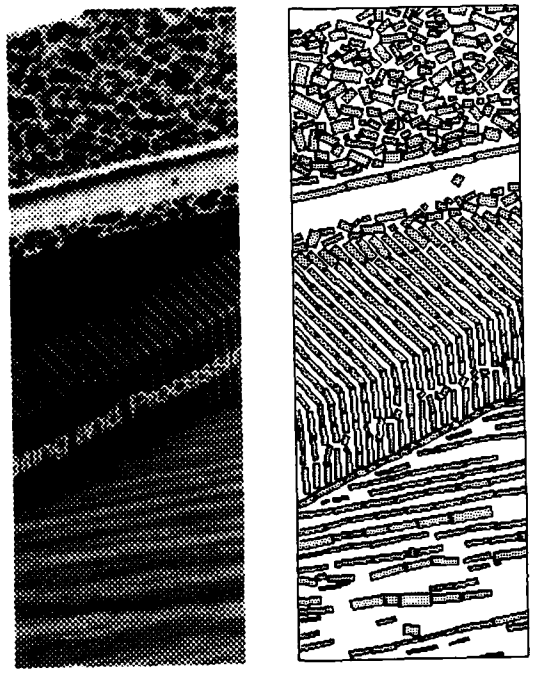
4 Why blobs?

We conjecture that texture in natural scenes can be represented using blobs (as detected by our algorithm) and their attributes.

By using blobs instead of edges, our conjecture resembles Julesz's; however, it is based on the analysis of natural, rather than synthetic, images. We claim that blobs are more useful than edges for representing texture for two reasons. First, attributes of blobs capture more information than attributes of edges. First order statistics of blob attributes account for higher order statistics of edge attributes. In particular, the average perpendicular distance between edges, a second order measure used by some (e.g., Kjell and Dyer, 1985) are captured by the average width of the blobs between them. Second, blobs provide a more appropriate resolution for representing the physical events which cause intensity changes. A blob can describe the approximate location, size, and orientation of an impression in a surface, even when its bounding contour (i.e., edges) is not well defined.

Interestingly, in the images which we have examined, dark blobs are useful more often than light ones as textons. Small impressions in the surface, such as spaces between threads in a fabric, or gaps between leaves on a tree manifest themselves as small dark regions in the image. Less light is reflected from such surfaces because of shadows cast by objects nearer the light source, or because of the oblique angle of these surfaces with respect to the viewer. Even surface markings, such as type on a page, are more often than not dark on light, resulting in isolated dark blobs.

If desired, termination points can be marked at locations which are the endpoints of single elongated blobs, while junctions can be marked where three or four blobs meet. However, it is not clear from natural images that these features are as relevant to the interpretation of texture as some psychophysical experiments might suggest. Even some of the boundaries perceived in the synthetic images of Julesz [1983] can be explained in terms of the basic blob attributes mentioned above [Voorhees, 1987].



(a) (b)

Figure 10: Blobs due to surface markings

(a) Close-up of of books on a wooden table top. (b) Strong blobs found at scale $\sigma = 1.5$ pixels.

5 Conclusion

We have shown how noise can be robustly estimated in natural images, enabling the automatic selection of thresholds for edge and blob detection. The method works for images consisting entirely of highly textured regions as well as those containing roughly uniform regions.

We have also described an algorithm for detecting intensity blobs in images, facilitating the construction of a raw primal sketch. In doing so we have answered the first question regarding the texton theory of vision: How can textons extracted from images of natural scenes?

We are currently studying the second obvious question: Once textons are detected, how, exactly, are texture boundaries identified? Marr [1976] suggested that first order statistics of attributes over regions are compared, but provides no algorithm. We are investigating two approaches.

The first method locates texture boundaries by using a non-linear smoothing operator to compute attributes over neighborhoods of textons, resulting in texton attribute arrays. This method, which is somewhat analogous to "preattentive" visual processing (described by [Julesz, 1983]), identifies gross changes in attributes which give rise to clearly perceived texture boundaries. Interestingly, these gross texture changes correspond to the type caused by physical discontinuities. We are currently doing experiments to determine, for each attribute, how large a difference should be considered significant.

The second method uses statistical tests to determine whether attributes of textons in regions on either side of a selected boundary are from different populations. This process, which may be analogous to "attentive" processing, can identify more subtle differences than the first method, differences which require scrutiny, and which do not usually correspond to physical discontinuities. These two methods are explained in detail in Voorhees [1987].

It should be noted that in the case of occlusion, density of textons is probably the most useful attribute. Two different surfaces are usually at different depths, and hence it is unlikely that both yield textons at a single scale of analysis. Texture boundaries based on blob attributes are more likely to occur at orientation discontinuities or in images where the depth of field is small.

Future research will address the third question: Once texture boundaries are detected, how can they be interpreted in terms of their physical cause? Riley [1981] offers some general observations, but more specific rules are needed. Given a description of the attribute distributions on either side of a boundary, one would like to ascertain the likelihood that the boundary is due to occlusion, a change in surface orientation, a surface marking, or a shadow.

6 References

Bracho, R. and A. C. Sanderson. "Segmentation of Images Based on Intensity Gradient Information," *Proc. CVPR-85*, pp. 341–347, 1985.

Canny, J. F. "Finding Edges and Lines in Images," MIT AI Lab, AI-TR-720, June 1983.

Horn, B. K. P. *Robot Vision.* Cambridge, MA: MIT Press, 1986.

Julesz, B. and J. R. Bergen. "Textons, The Fundamental Elements in Preattentive Vision and Perception of Textures," *Bell System Technical Journal*, Vol. 62, No. 6, July–Aug. 1983, pp. 1619-1645.

Kjell B. P. and C. R. Dyer, "Edge Separation and Orientation Texture Measures," *Proc. CVPR-85*, pp. 306–311, 1985.

Mahoney, J. V. "Image Chunking: Defining Spatial Building Blocks for Scene Analysis," M.S. Thesis, M.I.T. Dept. of E.E.C.S., Jan. 1987.

Marr, D. "Analyzing Natural Images: A Computational Theory of Texture Vision," *Cold Spring Harbor Symposia on Quantitative Biology*, Vol. XL, pp. 647–662, 1976.

Marr, D. *Vision*, San Francisco: W. H. Freeman, 1982.

Marr, D. and E. Hildreth, "Theory of Edge Detection," *Proc. Royal Society of London*, Vol. B-207, pp. 187-217, 1980.

Riley, M. D. "The Representation of Image Texture," MIT AI Lab AI-TR-649, Sept. 1981.

Voorhees, H. "Finding Texture Boundaries in Natural Images," M.S. Thesis, M.I.T. Dept. of E.E.C.S., Jan. 1987.

Witkin, A. "Scale-Space Filtering," *Proc. IJCAI-83*, pp. 1019-1022.

RECONSTRUCTION OF SURFACES FROM PROFILES

Peter Giblin *
Department of Mathematics
University of Massachusetts, Amherst, MA 01003

Richard Weiss †
Dept of Computer and Information Science
University of Massachusetts, Amherst, MA 01003

ABSTRACT

This paper presents an algorithm for computing a depth map for a smooth surface from a sequence of profile curves. The algorithm requires that the viewing directions be coplanar. In addition formulae are derived for computing directly the Gauss and mean curvatures without first computing a depth map. We have used the algorithm to reconstruct curves from their profiles with a high degree of accuracy from synthetic, noise-free data.

1. INTRODUCTION

We can tell a lot about the shape of an object from a single profile, and with multiple views we can often determine the shape uniquely. In this paper we analyze this reconstruction process mathematically and derive an algorithm to produce a depth map. For some applications it may not necessary to produce a depth map at all (for example in recognition problems). The Gauss and mean curvatures may be sufficient by themselves to solve this problem, and in any case it will be useful to decompose surfaces into patches according to whether they are convex, concave, hyperbolic, parabolic, or planar (Besl and Jain [1], Brady et al. [2], Ferrie and Levine [4]). This gives a method for describing surfaces of objects and a basis for matching with standard surfaces such as spheres, planes, and cylinders. Koenderink and van Doorn [7] have derived a formula which could be used to compute the Gauss curvature from a sequence of intensity images without depth. We have extended these results to produce formulae for the mean curvature, principal curvatures, and principal directions.

The use of profiles for the recognition and description of surfaces has been explored by many people (Koenderink and van Doorn [7], Callahan and Weiss [3], Hoffman and Richards[5]), but most investigations have not combined information from multiple views. In general, there is no way to identify a point on one profile with a corresponding point on a profile from a different view since for smooth surfaces they will not have any points in common. In fact, most stereo algorithms which are based on correspondence find the most similar point and assume it is the same. However, if the camera motion is known, then there is a method to identify points on two different profiles. In our work, we have restricted the camera to planar motion, so that planes parallel to the plane of motion induce a correspondence between the profiles. However, it is possible for the profile to change qualitatively between views, and in order to understand this, we have analyzed the analogous problem for a curve in the plane. These transitions create ambiguities in the reconstruction process. The criterion used to resolve this ambiguity is that the most likely solution is the one which minimizes the change in depth between adjacent views.

The mathematical approach to this problem is that a smooth surface without inflection points is the envelope of all of its tangent planes. However, there are two problems with this; how to compute the envelope of a family of planes and how to handle inflection points. With the assumption of planar camera motion, we have been able to reduce the envelope of planes problem to that of computing the envelope of a family of lines in a plane, which we were able to solve. The problem of inflection points requires interpolation of the surface. We have a simple approach to this which would work in most cases, and we plan to extend this to polyhedral surfaces where every points is either an inflection point or a crease point.

In Section 2, we give the mathematical framework for discussing profiles. Then for simplicity in Section 3, we take the case of reconstructing a plane curve. In Section 4, this is generalized to surfaces. In Section 5, we present the

* The work was done while the author was on leave from Dept of Pure Mathematics, University of Liverpool L693BX, England, and was a Visiting Professor of Mathematics at the University of Massachusetts. He also received generous support from Five Colleges Inc.

†The work of this author was supported by the Air Force Office of Scientific Research under grant F49620-83-C-0099, by the Defense Advanced Research Projects Agency under contracts N00014-82-K-0464 and DACA76-85-C-0008, and by the National Science Foundation under grant DCR-8318776.

experimental results using simulated data.

2. THE PROFILES OF A SURFACE

Let u be a unit vector in 3-space $\mathbf{R}^3$ and let M be a smooth surface in $\mathbf{R}^3$. We regard u as defining a viewing direction. On the surface M there is a locus of points p for which the tangent plane contains the viewing direction. This locus of points is called the *critical set* corresponding to u. If this curve is projected parallel to u onto the viewing plane through the origin which is perpendicular to u, the image is called the *profile* (Figure 1). For future reference we note the following standard facts about critical sets and profiles:

- The critical set is a smooth curve at p unless p is a parabolic point and u is the asymptotic direction.
- The profile near q in the viewing plane arising from p on the critical set is a smooth curve unless the viewing direction is an asymptotic direction.
- When the critical set is smooth at p, its tangent direction is parallel to the viewing plane if and only if the viewing direction is a principal direction at p.
- When the profile is smooth at q, its tangent at q is always in the tangent plane to the surface at p.

The Gauss curvature can be computed from the profiles. Consider a point p of the critical set. The viewing direction at p together with the normal to the surface there determine a plane, whose intersection with the surface is a smooth curve. The curvature of this curve at p on the critical set is called the radial curvature. The projection of p onto q lies on the profile, and the curvature of the profile at q is called the transverse curvature. Koenderink and van Doorn [7] showed that the Gauss curvature is the product of the transverse curvature and the radial curvature. Brady et al. [2] has also given an independent proof of this fact. The curvature of the profile can be computed directly from an image, and the radial curvature could be computed from a sequence of images.

3. RECONSTRUCTING PLANE CURVES

Consider a smooth plane curve C, which we usually take to be closed, so that it is given by a parametrization $\gamma(t) = (X(t), Y(t))$ such that the derivative is never zero, i.e. $X'(t)$ and $Y'(t)$ are never zero for the same t. As shown in Figure 2, u is the *viewing direction*. The *viewing line* is the line perpendicular to u through the origin. The profile in this context is the set of points on the viewing line for which the tangent line is parallel to u. The position of a profile point can be expressed as $w \cdot (\sin\theta, -cos\theta)$, where w is the signed distance from the origin, O, to the profile point. For example, in Figure 2, there are three points for which $w > 0$ and one for which $w < 0$. Note that if θ produces a value w, then $\theta + \pi$ produces $-w$. Thus we restrict to $0 \leq \theta < \pi$. The collection of all profile points forms a curve in the plane classically called the *pedal curve* of C with respect to O. If θ changes in such a way that a pair of values of w is annihilated and disappear, or a pair is created, then the pedal curve has a singular point (usually a cusp). This happens when u passes through the direction of an inflexional tangent to C, as shown in Figure 3. In a neighborhood of that point, w cannot be a smooth function of θ. In general, there will be parts of C for which w is a function of θ for some range of values of θ. Suppose C_1 is such a subset of C, then the following proposition states that C_1 can be reconstructed from the function $w = w(\theta)$. Any part of the curve which has no inflexions satisfies this property. Another way to say this is that the Gauss map of C_1 has an inverse, i.e. for each direction on the unit circle there is at most one point of C_1 whose normal points in that direction. Such curves possibly with singularities have been studied by Langevin, Levitt, and Rosenberg [9] are called *herissons* (hedgehogs)

Proposition 1 *1. The curve C_1 consists of points*

$$x = w\sin\theta + w'\cos\theta$$

$$y = -w\cos\theta + w'\sin\theta$$

and $w' = dw/d\theta$ is the distance from the profile point to the corresponding point of C_1

2. The radius of curvature of C_1 at the point corresponding to θ is $w + w''$. (Here C_1 is oriented by increasing θ.)

proof: The line through the profile point $w \cdot (\sin\theta, \cos\theta)$ parallel to the viewing direction $u = (\cos\theta, \sin\theta)$ has the equation

$$((x, y) - (w\sin\theta, -w\cos\theta)) \cdot (\sin\theta, -\cos\theta) = 0$$

i.e.

$$x\sin\theta - y\cos\theta = w \tag{1}$$

The curve C_1 is the *envelope* of the lines obtained by eliminating θ between equation (1) and the following equation:

$$x\cos\theta + y\sin\theta = w' \tag{2}$$

This gives the unique solution (x, y) in part 1 of the Proposition. Rewriting this as

$$(x, y) = w(\sin\theta, -\cos\theta) + w'(\cos\theta, \sin\theta)$$

proves that w' is the distance. The formula for the radius of curvature for a curve is given by:

$$\rho = (x'^2 + y'^2)^{3/2}/(x'y'' - x''y')$$

Differentiating the equations in part 1 and substituting into the formula for ρ gives part 2 of the Proposition.

It is not possible for $w + w''$ to be zero while C_1 is smooth: zero radius of curvature is only possible at a cusp of a curve. On the other hand $w + w''$ can tend to infinity, making the curvature tend to zero (which is an inflexion). But at an inflexion w is no longer a function of θ so it is not in C_1.

Examples of these two cases are $w = \theta^3$ and $w^2 = \theta^3$ respecively (see Figure 4). For the former, $x = 3\theta^2$+higher order terms, and $y = 2\theta^3$+ higher order terms. Thus C_1 has a cusp and the pedal curve has an inflexion at $\theta = 0$. For the latter, $x = \pm\frac{3}{2}\theta^{1/2}$+ higher order terms, and $y = \pm\frac{1}{2}\theta^{3/2}$+ higher order terms. Thus, C_1 has an inflexion and the pedal curve has a cusp at $\theta = 0$.

In practice if we start with a curve C, and measure its profile data, i.e. for each viewing direction $u = (\cos\theta, \sin\theta)$ for some range of values for θ we obtain the various values of w for the profile points. Some values of θ will give more values of w than others, unless C has no inflexions, in which case each value of θ will have two values of w. Starting at some value $\theta = \theta_0$, we choose one of the corresponding values to be w_0 and increase θ following w as a function of θ until an inflexion is encountered. This is detected by a change in the number of w-values. In fact θ_0 can be chosen so that it immediately follows after an inflexion. It turns out that one only needs to consider the case when the number of w-values decreases by two. The parts of the curve C lying between inflexions can be reconstructed using part 1 of the Proposition. The computational problem to be solved is this: having chosen w_0 for $\theta = \theta_0$, what value corresponds to $\theta_0 + \delta\theta$? It is not sufficient to simply choose the closest value of w. Since it is possible for the pedal curve to have crossings (See Figure 5), there is the danger of starting on one branch and switching to the other. Note that we are approximating the function $w = f(\theta)$ at discrete points, and we have assumed that this function is continuous since C is smooth. In addition, since w' is the distance from the viewing line to the point of tangency on the curve, we also want w' to be continuous. This can be viewed as a constraint on choosing successive values of w such that w'' is minimized.

4. PERSPECTIVE PROJECTION OF CURVES

There is no essential difference in the mathematics of reconstructing curves from profiles obtained from perspective projection. We derive the formulas here for completeness. Assume that the curve C is contained in a circle of radius r. From each point A on the circle we define the viewing line to be the line parallel to the tangent to the circle at A and a distance d away from it. Each tangent line to C through A will intersect the viewing line at a profile point. (See Figure 6) The profile points are identified by their distance w in a counterclockwise direction from the origin of the viewing line, which is the closest point to A. The values of w for θ and $\theta + \pi$ are no longer directly related.

Proposition 2 *For regions of the curve in which w is a function of θ,*

$$x = \frac{rwd\sin\theta + rw^2\cos\theta + rw'd\cos\theta}{d^2 + w^2 + dw'}$$

$$y = \frac{rwd\cos\theta + rw^2\sin\theta + rw'd\sin\theta}{d^2 + w^2 + dw'}$$

Thus in this case also the parts of C between inflexions can be recovered from a knowledge of the function w. However, in this case the formula for the curvature is rather more complicated.

5. SURFACES

We would like to reconstruct a surface from its profiles in a way analogous to the method used for curves in the previous section. The situation for surfaces differs in two respects: each profile is a curve and there is a two-parameter family of viewing directions which as unit vectors are points on the sphere S^2. We seek to find all the tangent planes to M, and for this purpose, it is sufficient (and necessary) to know the profiles for a "great circle" of viewing directions, i.e. all directions which lie in a plane. Consider the tangent plane to M at p. The unit tangent vectors in this plane form a great circle on S^2, and this circle intersects the great circle of viewing directions. Hence some tangent line to M at p is in the viewing direction u, so that p contributes a point q to the profile for the viewing direction u. Of course, for opaque surfaces the situation is complicated by occlusion, and it is possible that for some points, none of the singular sets would be visible.

How do we find the tangent plane to M at p? One line in that plane is, of course, the line through q parallel to u. Another line is the *tangent line to the profile* at q (see Figure 7). If the profile is singular at q then the tangent line is interpreted as a limit of tangent lines at nearby smooth points of the profile. If the profile has a crossing at q, then both branches will contribute a tangent plane to M but at different points p of M ([7], [6], [3]).

We shall now reconstruct M from the profiles corresponding to a circle of viewing directions (except where the tangent plane is parallel to the plane of the circle of viewing directions). The calculation which follows is local in that it assumes a parametrization of profiles near each point.

Consider a family of viewing directions, $u = (0, \cos\theta, \sin\theta)$ and corresponding viewing planes $y\cos\theta + z\sin\theta = 0$. Each viewing plane will contain a profile of M; we use orthonormal coordinates (x, w) in the viewing plane, where the w-axis is in the direction $(0, \sin\theta, \cos\theta)$ and the x-axis is the same for all of the planes. A point (x, w) in a viewing plane then is the point

$$x(1,0,0) + w(0, \sin\theta, -\cos\theta) = (x, w\sin\theta, \quad w\cos\theta) \quad (3)$$

in (x,y,z) space. In practice it is the numbers (x, w) which will be measured from an image.

In general, the profile will be be a curve with isolated cusps, and we present the theory for reconstructing the surface from smooth points of profiles. The case of those cusp points is still under investigation. In practice, this should not be a problem since one can compute the surface at points in a neighborhood of a cusp where the profile is smooth. Assume that the profiles have the form $w = w(x,\theta)$ over a range of values of θ and x, where $w(x,\theta)$ is a smooth function. Thus we assume that over some range of values of x and θ the profiles are smooth and not tangent to the w-axis. So starting with a point on a given profile of M, we choose an axis of rotation which is not parallel to the normal to the profile at that point. It is intuitively reasonable that if the viewing direction remains in the tangent plane, no additional information will be obtained.

Now consider a fixed x and θ, i.e. a fixed point on a particular profile curve. The tangent plane to M determined by this x and θ passes through $(x, \sin\theta, -w\cos\theta)$ and contains the directions:

$(0, \cos\theta, \sin\theta)$ (the viewing direction)

$(1, w_x\sin\theta, -w_x\cos\theta)$ (tangent to the profile)

where $w_x = \partial w/\partial x$. The equation of the plane is therefore

$$w_x X - \sin\theta Y + \cos\theta Z = xw_x - w \qquad (4)$$

where we temporarily use (X,Y,Z) as current coordinates in $\mathbf{R}^3$ to avoid the double use of x.

Remark: If the profiles are parametrized as $x = x(t,\theta)$, $w = w(t,\theta)$ for some parameter t, then the tangent plane is

$$w_t X - x_t \sin\theta Y + x_t \cos\theta Z = xw_t - x_t w \qquad (5)$$

The surface M is the *envelope* of all the tangent planes described by (4), that is we obtain the point (X,Y,Z) of M corresponding to (x,θ) by eliminating x and θ between (4) and its derivatives with respect to x and θ, viz.

$$\partial/\partial x: \quad w_{xx} X = xw_{xx} \qquad (6)$$

$$\partial/\partial\theta: \quad w_{x\theta} X - \cos\theta Y - \sin\theta Z = xw_{x\theta} - w_\theta \qquad (7)$$

This amounts to finding the intersection of three tangent planes given by (x,θ), $(x+\delta x,\theta)$, and $(x,\theta+\delta\theta)$. The intersection of these three planes will approach the point on M in the limit as δx and $\delta\theta$ go to zero. This runs into problems when one or the other direction produces a stationary tangent plane. According to equation (6), $X = x$ (the line through a profile point in a viewing direction is always in a plane X = a constant), but (6) also indicates that if the profile has an inflexion ($w_{xx} = 0$), then there will be problems distinguishing one tangent plane to M from the "next". This is similar to the case for curves in which the envelope of tangent lines to a plane curve with an inflexion contains the whole inflexional tangent line. In fact (4), (6), and (7) determine a unique point (X,Y,Z) if and only if $w_{xx} \neq 0$, namely the point

$$f(x,\theta) = (x, w\sin\theta + w_\theta\cos\theta, \quad w\cos\theta + w_\theta\sin\theta) = M \quad (8)$$

Note that w_θ is the distance from a profile point to the corresponding point on M.

Remark In the general case where w is not always a function of x, we can parametrize the family of profile curves as follows:

$$x = x(t,\theta)$$
$$w = w(t,\theta)$$

In this case, except where x_t is zero, equation (8) becomes

$$f(t,\theta) = (x, w\sin\theta, -w\cos\theta) + \begin{pmatrix} x_\theta w_t + x_t w_\theta \\ x_t \end{pmatrix} (0, \cos\theta, \sin\theta) \qquad (9)$$

and $(-x_\theta w_t + x_t w_\theta)/x_t$ is the distance from a profile point to a point on M.

The formula (8) tells us how to reconstruct M from its profile data, as long as w can be expressed as a function of x and θ. The condition for f to give a smooth piece of surface (i.e. the condition that the differential of f has rank 2) is $w + w_{\theta\theta} \neq 0$; note that this also arose in the case of curves above.

On M at any point $f(x,\theta)$, we have coordinate directions corresponding to:

$$\partial f/\partial x = f_x = (1, w_x\sin\theta + w_{x\theta}\cos\theta, \quad w_x\cos\theta + w_{x\theta}\sin\theta)$$

where f_x is in the direction of the *critical set* and f_θ is in the *viewing direction* so they are not in general orthogonal. Nevertheless, they are *conjugate* with respect to the second fundamental form (see Figure 8). Note that f_x and f_θ will not coincide at a smooth point of the profile. Geometrically, this means that with respect to the ellipse determined by this quadratic form, each direction is tangent to the ellipse at the point of intersection by the axis determined by the other. Algebraically, the matrix associated with the second fundamental form is diagonal with respect to the basis of these two directions and (using $n = (\ w_x, \sin\theta, \quad \cos\theta)/(1+w_x^2)^{1/2}$ as the unit normal to M) can be written as:

$$\begin{pmatrix} \frac{w_{xx}}{(1+w_x^2)^{1/2}} & 0 \\ 0 & \frac{w+w_{\theta\theta}}{(1+w_x^2)^{1/2}} \end{pmatrix}$$

Thus, it is possible to derive simple formulae the Gaussian and mean curvatures of M in terms of the profile data w. As noted above, the consequent formula for the Gaussian curvature as the product of the radial curvature, κ_r and the transverse curvature, κ_θ, was known,but the mean curvature cannot be expressed in terms of only these two.

6. RELATIONSHIP BETWEEN SECTIONAL CURVATURES AND THE SURFACE CURVATURES

There are three curvatures which enter into the equations for the surface curvatures.

κ_c is the curvature of the profile or the radial curvature. Its formula is

$$\kappa_c = w_{xx}/(1 + w_x^2)^{3/2}$$

κ_x is the sectional curvature of M in the f_x direction (the direction of the tangent to the critical set). We find (see below) that

$$\kappa_x = w_{xx}/|(1 + w_x^2)^{1/2}(1 + w_x^2 + w_{x\theta}^2)|$$

κ_θ is the sectional curvature of M in the f_θ direction, which is the viewing direction u when $w + w_{\theta\theta} > 0$ and $-u$ when $w + w_{\theta\theta} < 0$. This is also known as the transverse curvature. We find (see below) that

$$\kappa_\theta = -1/|(1 + w_x^2)^{1/2}(w + w_{\theta\theta})|$$

Proposition 3 *1. The Gauss curvature K and the mean curvature H of M at $f(x,\theta)$ are given by*

$$K = -w_{xx}/|(1 + w_x^2)^2(w + w_{\theta\theta})| = \kappa_c\kappa_\theta$$

$$H = \frac{w_{xx}(w + w_{\theta\theta}) - 1 - w_{x\theta}^2}{2(w + w_{\theta\theta})} - \frac{1}{2}\kappa_c\left(1 + \frac{\kappa_\theta}{\kappa_x}\right)$$

2. $w_{x\theta} = 0$ if and only if the viewing direction is a principal direction on M at the corresponding point. In this case f_x and f_θ are the principal directions, $\kappa_c = \kappa_x$, and $H = \frac{1}{2}(\kappa_c + \kappa_\theta)$.

proof: The computation of the Gauss and mean curvatures from a local parametrization f of M can be found in O'Neill [8]. We can obtain f_x and f_θ from equation (8) and compute the first fundamental form as

$$\begin{pmatrix} 1 + w_x^2 + w_{x\theta}^2 & w_{x\theta}(w + w_{\theta\theta}) \\ w_{x\theta}(w + w_{\theta\theta}) & (w + w_{\theta\theta}) \end{pmatrix}$$

This together with the second fundamental form given above allows us to compute the surface curvatures. Thus K and H can be expressed in terms of the curvatures $\kappa_c, \kappa_x, \kappa_\theta$:

The shape operator, which is the derivative of the Gauss mapping, referred to the basis f_x, f_θ is:

$$\frac{1}{(1 + w_x^2)(w + w_{\theta\theta})}\begin{pmatrix} w_{xx}(w + w_{\theta\theta}) & -w_{xx}w_{x\theta} \\ w_{x\theta}(w + w_{\theta\theta}) & -(1 + w_x^2 + w_{x\theta}^2) \end{pmatrix}$$

The principal directions are the eigenvectors of this matrix. One can see that, given the assumption that $w + w_{\theta\theta}$ is nonzero, the matrix is diagonal if and only if $w_{x\theta} = 0$. The principal curvatures are the eigenvalues, and the Gauss curvature is the product of the eigenvalues and the mean curvature is their sum. Thus, when the matrix is diagonal, f_x and f_θ are the principal directions and they are orthogonal.

When the profile has a cusp, the curvatures K and H will be the limits of the values of the formulae at corresponding smooth points near it. However, K for example cannot be expressed as $\kappa_c\kappa_\theta$, since κ_c is infinite and κ_θ is zero. Finding a formula to replace this is a goal of current investigation.

7. EXPERIMENTAL RESULTS

In order to determine the potential accuracy of the algorithm for reconstructing curves, several experiments were performed. Synthetic data was used to generate profile points for the various values of θ. Figure 8. shows a picture of a curve with two inflexions and the pieces of the curve which are reconstructed from the data.

In principle, one could start with the profile data and trace the curve from start to finish, reversing direction at inflexion points, always choosing the value of w which minimizes w''. However, there is no sure test on the values for w which will guarantee that a point is an inflexion point. It is possible to find all of the tangent directions for the inflexion points, which can be called *flex directions*. These are the directions at which the number of tangent lines changes. We use these flex directions to break up the profile data into *heurisson* segments for which w is a function of the viewing direction. The current algorithm constructs arcs corresponding to the *heurisson* segments between inflexion points.

As a practical point, one can choose the place to start drawing the curve to be any w value after a flex direction. Once the pieces of the curve between the inflexions are drawn, they can be linked together with straight lines based on proximity of endpoints (assuming one has started with a closed curve. In theory one would like to minimize the integral of the absolute value of w''. In the current algorithm we only apply this criterion locally to choose the values of w sequentially. It may be necessary to apply standard relaxation techniques when dealing with real data.

The algorithm can be extended easily to surfaces, and we intend to apply this to the problem of model acquisition from physical prototypes.

8. CONCLUSION

We have given a procedure for reconstructing surfaces from a sequence of profiles. In addition we have given a formula for mean curvature in terms of the profile data which extends the similar result for the Gauss curvature. This makes it possible to compute both of these curvatures without first computing a dense depth map. We have used this algorithm on synthetic, noise-free data to reconstruct

curves with a high degree of accuracy, and we plan to test it on real data.

ACKNOWLEDGEMENTS

We would like to thank James Callahan for his many helpful conversations and suggestions.

REFERENCES

[1] Besl, P.J. and Jain,R.C., "Segmentation Through Symbolic Surface Description" *Proceedings CVPR86*, Miami, June 1986, pp. 77-85.

[2] Brady,M., Ponce,J., Yuille,A., and Asada,H., "Describing Surfaces", *Proceedings of 2nd International Symposium on Robotics Research*, Hanafusa and Inoue (Eds.), MIT press, Cambridge, Ma.

[3] Callahan, J. and Weiss, R., "A Model for Describing Surface Shape", *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, San Francisco, June 1985, pp. 240-245.

[4] Ferrie, F.P., and Levine, M.D., "Piecing Together the 3D Shape of Moving Objects: An Overview", *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, San Francisco, June 1985, pp. 574-584.

[5] Hoffman, D.D. and Richards, W., "Parts of Recognition", MIT AI Memo 732, Dec. 1983.

[6] Kergosien, Y.L., "La famille des projections orthogonales d'une surface et ses singularities", *C.R. Acad. Sc. Paris*, 292 , pp. 929-932, 1981.

[7] Koenderink, Jan J., "What Does the Occluding Contour Tell Us About Solid Shape?", *Perception*, vol 13, 1984, pp. 321-330.

[8] O'Neill, B., *Elementary Differential Geometry*, Academic Press, New York, 1966.

[9] Langevin, R., Levitt, G., and Rosenberg, H., "Herissons et Multiherissons", preprint.

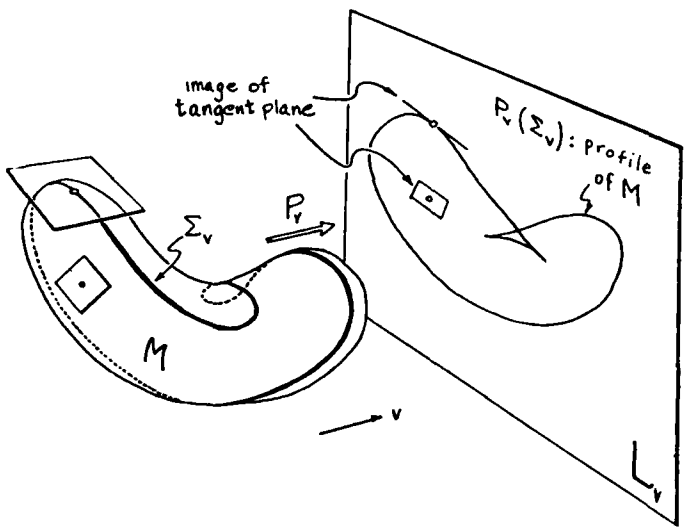


Figure 1: The critical set Σ_M and profile of the projection of M

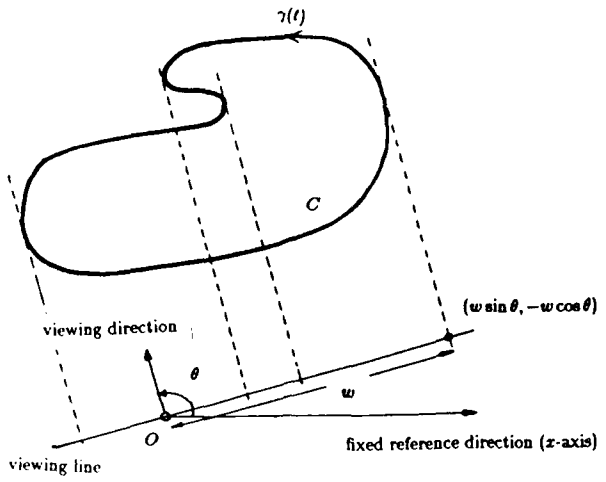


Figure 2: The profile of a plane curve is a set of points on the viewing line

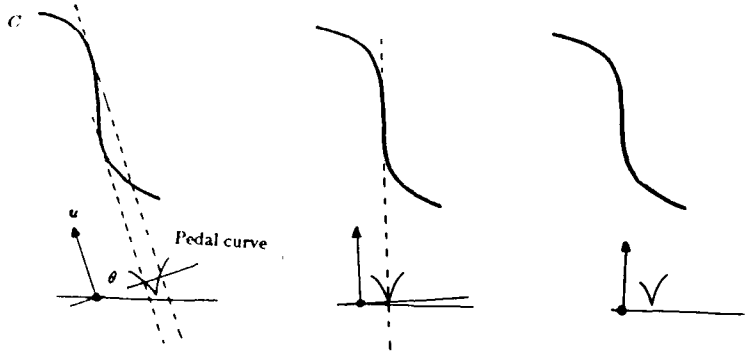


Figure 3: An inflexion in C produces a cusp in the pedal curve

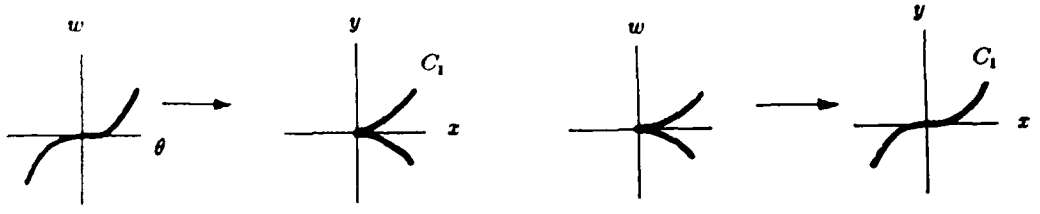


Figure 4: Examples of pedal curves for a cusp and an inflexion

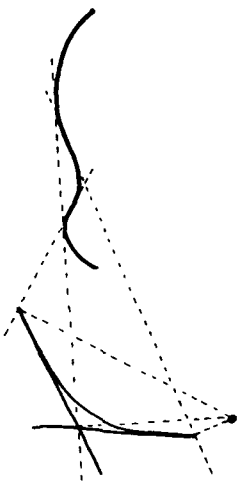


Figure 5: A point where the pedal curve crosses itself

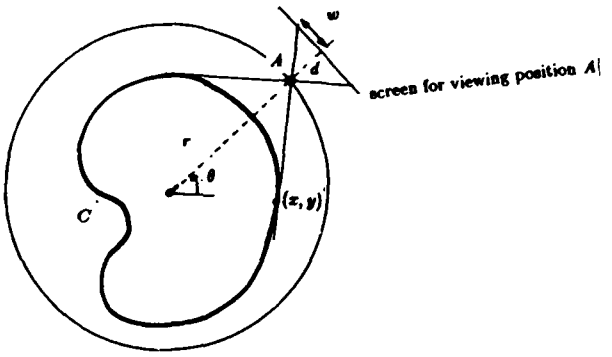


Figure 6: The viewing line for perspective projection

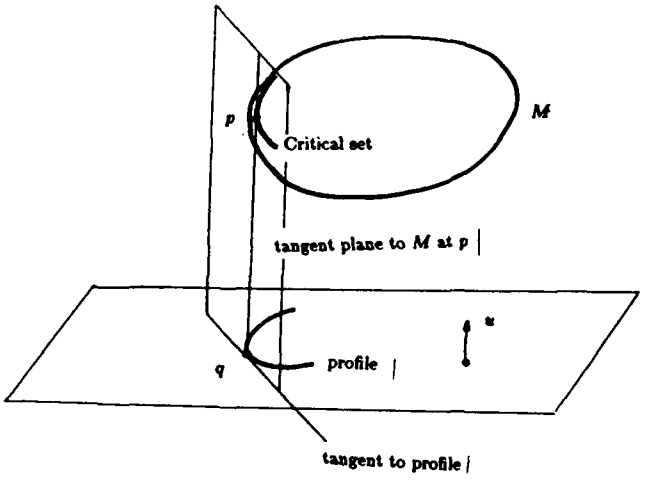


Figure 7: The tangent plane at p contains the viewing direction and a vector parallel to the tangent to the profile curve at q

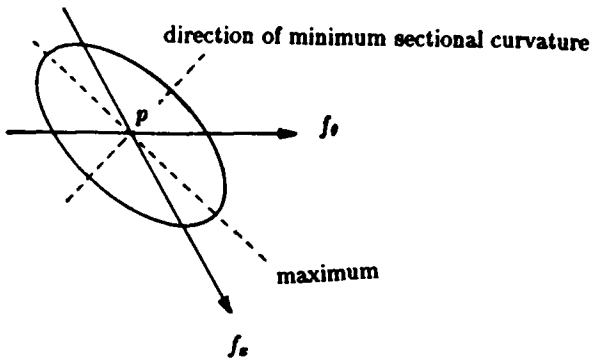


Figure 8: The viewing direction and the tangent to the critical set are conjugate

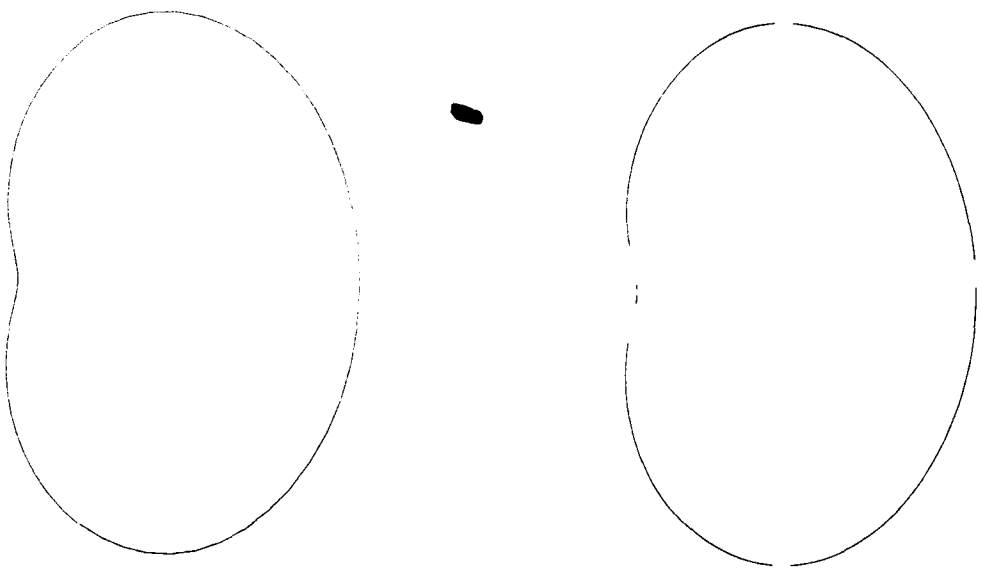


Figure 9: a. A drawing of the limacon from initial data. b. The reconstruction of the limacon.

USING OCCLUDING CONTOURS FOR OBJECT RECOGNITION

Willie Lim

Massachusetts Institute of Technology
Artificial Intelligence Laboratory
Cambridge, MA 02139

Abstract

This paper describes an algorithm, in the area of symbolic high level vision, that is implemented on the Connection Machine. The domain chosen is the rocks world where objects are rocks or mountains. These objects are represented using qualitative surface models. The algorithm takes a library of such models and loads them all into the Connection Machine. Candidate images are processed to extract occluding contours which are then matched in parallel to all the library models. The algorithm produces all the views of the models that match the occluding contours.

1 Introduction

In the rocks world, objects are hard to describe using quantitative surface models. For this reason, a qualitative model is used. In this model an object is partitioned into surface patches. Each surface patch is qualitatively typed according to its surface shape e.g. flat or doubly convex. A graph is constructed with the nodes representing the surface patches and arcs representing the neighborhood relationships between the patches. The arrangement of surface patches vary from object to object in an unpredictable way. Thus it is very unlikely that two rocks will have an identical arrangement of surface patches. This is unlike man-made objects where objects can be made to have identical shapes. It is the unpredictable nature of such surface properties in the rocks world that enables the physical constraints imposed by these qualitatively described surface patches to be effective for shape recognition.

This paper describes how a convex object in the rocks world can be recognized using its occluding contour derived from an image of the object. The emphasis here is illustrating how the Connection Machine [3] can be used for object recognition, and in particular, in shape recognition in the rocks world. A survey of Connection Machine algorithms for other applications is given in [4]. The algorithm presented in this paper is implemented in *LISP [1].

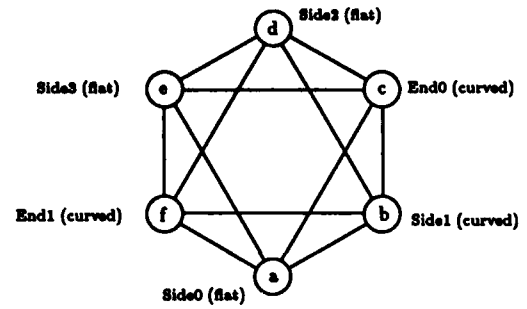
The paper is organized as follows. The various terms used in the paper are defined in Section 2. Several constraints are used in the algorithm. They are discussed in Section 3. Section 4 describes the algorithm and the conclusion of the paper is given in Section 5.

2 Occluding Contours of Rocks

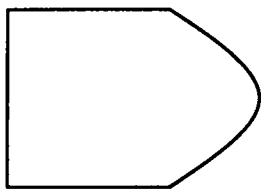
The surface of a rock is partitioned into surface patches, each of which is assigned a *surface type* that qualitatively describes the overall shape of the surface patch. For example a surface patch that appears flat will be typed as flat. Such characterization of surface patches are relative to the neighboring surface patches. Hence it is possible for a curved surface patch on one rock to be classified as flat on another rock. The matching algorithm presented in this paper exploits the constraints imposed on each other by these surface patches.

Using the surface patches, a rock is modeled as a graph termed a *model graph*. Each node in the model graph corresponds to a surface patch and each arc represents the adjacency (i.e. neighborhood) relationships of the surface patches. The recognition task starts with an *occluding contour* i.e. a closed curve that marks the discontinuity in depth between the object and the background [6]. The occluding contour is partitioned into segments which are assigned types called *contour types* e.g. straight or (doubly) curved. For exposition in this paper, the number of surface types is limited to two—flat or curved. Similarly the number of contour types is limited to two. In practice the number of surface types can be increased to six—doubly curved convex, doubly curved concave, saddle shaped, singly curved convex, singly curved concave, and flat. The number of contour types can be increased to three—convex, concave and straight.

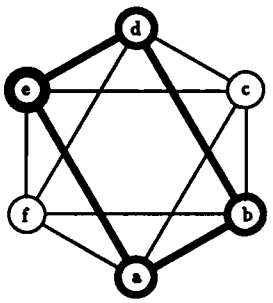
The sequence of symbols representing the contour types of an occluding contour traversed in a given direction is termed a *contour string*. For a surface patch to match a symbol of the contour string, the surface patch must have a



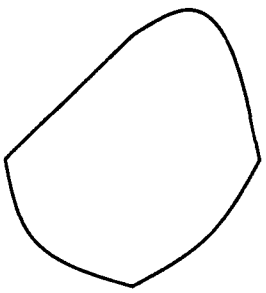
(a) Model graph



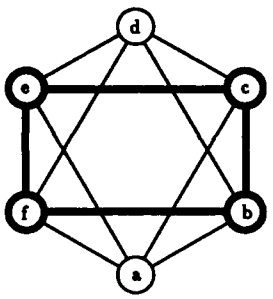
(b) Occluding contour for end view



(c) Matching nodes for end view



(d) Occluding contour for top view



(e) Matching nodes for top view

Figure 1: An example.

surface type that is consistent with the contour type represented by the symbol. Thus if the segment of the occluding contour is straight, it can only match a flat surface patch in the world where surface types are restricted to be doubly curved or flat. This has to be true as each occluding contour segment is due to a surface patch that grazes the line of sight. Thus if the occluding contour segment is curved, the surface must be curved. Furthermore surface patches matching adjacent symbols in the contour string must also be adjacent in the model graph.

Since an occluding contour is a closed curve, the string of nodes that matches the contour string must form a cycle. Such a string of nodes is termed a *node string*. It partitions the model graph into two halves which correspond to the two complementary "front" and "back" views of the object that produce the same occluding contour. To distinguish between the two views, further information about the internal structure of the object has to be derived from the image. Such information might be the number and types of the surface patches inside the occluding contour.

Throughout this paper, the example shown in Figure 1 is used for illustrating how the algorithm works. Figure 1(a) shows the model graph for a convex object with six faces, three of which are curved. An example of such an object is a rectangular block (e.g. a generalized cone 2 with a square cross-section and a straight spine) with two of its ends (surface normals parallel to the axis or direction of sweep) and one side (surface normal perpendicular to the axis) curved instead of flat. The curved side is also adjacent to the bottom (and top) side of the object. Suppose that a view of the object produces the occluding contour shown in Figure 1(b). One of the contour string representing the contour would be the string of contour types—(*curved*, *straight*, *straight*, *straight*). Other possibilities are rotations and reversals of this string. The occluding contour corresponds to viewing the object from a point along its axis, at a sufficient distance away from the object. The matching nodes (and arcs) are shown in dark lines in Figure 1(c). Another occluding contour is given in Figure 1(d). The contour string for this view is (*straight*, *curved*, *curved*, *curved*). It corresponds to viewing the object from the top or bottom. Figure 1(e) shows the matching nodes (also in dark lines).

3 The Constraints

The algorithm generates all possible matches for a given contour string i.e. all possible node strings (and hence views). As the algorithm progresses the number of possibilities is reduced by applying five constraints.

The first of these constraints is the *type consistency constraint*: the contour type of the contour segment being matched must be consistent with the surface type of the surface patch. This means that a flat contour segment can only match a flat surface and a curved segment a curved surface.

The second constraint is the *adjacency constraint*: if a node matches the position, i, where $0 \leq i \leq L-1$ and L is the length of the contour string, it must be adjacent to nodes whose types are consistent with the contour types at positions $(i-1) \bmod L$ and $(i+1) \bmod L$. In other words the adjacency constraint involves the application of

the type consistency constraint to the node and two of its neighbors given that the node is at position i. Thus if the node x matches position i and its neighboring nodes, w and y, match positions $(i - 1) \bmod L$ and $(i + 1) \bmod L$, respectively, then wxy is a possible substring of the node string. This constraint and the type consistency constraint are applied at the initialization phase of the algorithm.

It is assumed that surface patches are not so large as to cover the object from one side to the other. This means that a third constraint be used. This is the *uniqueness constraint*: nodes in the node string must be unique. This means that as the node string is being built, the nodes being added must be new ones.

Since an occluding contour is a closed curve, there is a fourth constraint called the *wrap around constraint*: the first and last nodes in the node string must be adjacent. This constraint is used at the end of the algorithm for eliminating illegal node strings.

The fifth constraint is the *two-neighbor constraint*: if the length of the contour string is greater than 3, each node in the node string cannot be adjacent to more than two other nodes in the string. Applying this constraint is easy. Just make sure that the node at position i does not have more than two neighbors in the node string. This constraint limits the number of views that can be handled by the matcher. For example if four surface patches meet at a corner such that when the object is viewed from directly above the corner, they produce an occluding contour, all the nodes in the corresponding node string (with $L = 4$) will be adjacent to the rest of the nodes in the string. If it is known that there is a corner inside the occluding contour, this constraint can be ignored. With this in mind, a more comprehensive version of the constraint is being developed to overcome this limitation. Information about the internal structure of the rock will have to be used. For example if the number and types of internal surface patches are known, the type consistency, adjacency, and uniqueness constraints can be applied to the nodes representing these surface patches. Quantitative information relevant to relative surface areas and orientations can also be used to further reduce the number of possibilities. Such information will be useful, for example, for inferring when surface patches occlude one another and hence cannot be in the same node string. Information about surface areas will also be useful in determining when the uniqueness constraint can be applied.

4 Matching a Model to an Occluding Contour

Model graphs are stored on the Connection Machine by assigning a node to each processor. Nodes within a model graph are uniquely numbered. Each processor contains a table mapping node numbers to processor addresses. The list of neighboring nodes and their surface types are also stored in each processor as a list of pairs of node number and surface type.

The algorithm starts by broadcasting the contour string to all processors in the Connection Machine. Each processor then goes through an initialization step which involves finding the positions in the contour string that the processor can map to. The corresponding initial substrings involving the node and two of its neighbors are also computed. Then the adjacency and uniqueness constraints are applied as shown below. Note the constraints are applied by calling functions that return booleans and concatenations are denoted by the symbol $\circ$. After initialization, every processors will have list of pairs of values i.e. $(position, substring)$. The substring component of these pairs is updated until the complete node strings are obtained.

```
foreach processor p
 foreach position i in the contour string
  if
type-consistency-constraint(contour-type(i),surface-type(p))
  then
   forall neighbors x, y of p
    if adjacency-constraint(contour-type((i - 1) mod L),
                            surface-type(x),
                            contour-type((i + 1) mod L),
                            surface-type(y))
     and uniqueness-constraint(x, p, y)
    then collect (i, x ∘ p ∘ y)
      in list-of-legal-substrings-for-node-string
    endif
   endforall
  endif
 endforeach
endforeach
```

In the rest of the matching algorithm the substrings are lengthened by "distance-doubling" or "pointer-jumping" [5,7]. That is, the substrings are lengthened by concatenating longer and longer substrings which are obtained by communicating with processors that are further and further away. The length of these substrings and the distance between processors doubled at each iteration. The substring currently being computed in a processor is divided into three parts—*left*, *center* and *right*. In the rest of this paper, this triple of substrings is represented as $(left, center, right)$.

Figure 2 illustrates the case for 17 linearly linked nodes (i.e. each node is adjacent to its immediate neighbors). Consider the node i in the middle of the column of nodes. At initialization, its left, center and right substrings are h, i and j, respectively. Similarly nodes h and j have the substrings (g, h, i) and (i, j, h), respectively. The substrings

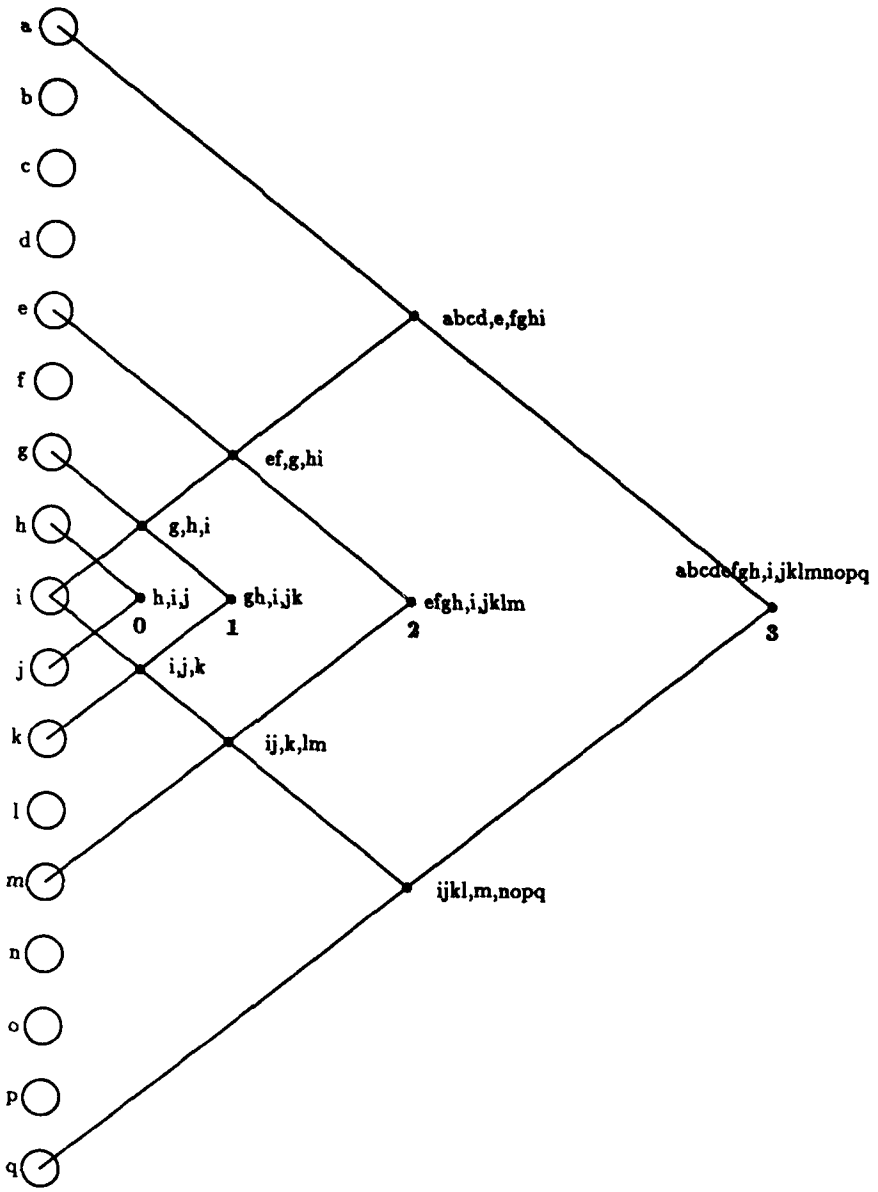


Figure 2: Building the node string.

are marked next to dark dots. The lines marks the leftmost and rightmost nodes of the left and right substrings, respectively, at the end of each iteration. Dot 0 represents the state of the substrings for node i just after initialization. At the first iteration, each node sends to the rightmost (leftmost) node of its right (left) substrings its current left (right) substrings. That is, node h sends its left substring g and node j its right substring k to node i. Node i updates its left and right strings to gh and jk, respectively, by concatenating the substring it receives from the right (left) node to its current right (left) substring. This is indicated in Figure 2 by dot 1 (to the right of dot 0). At the same time, node g has the substrings (ef, g, hi) (the dark dot vertically above dot 1) by receiving substrings from nodes e and i. Similarly, node k has the substrings (ij, k, lm) (the dark dot vertically below dot 1) using information sent to it by nodes i and m. At the second iteration, node i receives information from nodes g and k resulting in the new substrings $(efgh, i, jklm)$ (see dot 2). Meanwhile nodes e and m update their substrings to $(abcd, e, fghi)$ and $(ijkl, m, nopq)$, respectively. At the third iteration, nodes e and m communicate with node i causing it to update its substrings to $(abcdefgh, i, jklmnopq)$ (see dot 3). It can be seen that the lengths of the left and right substrings double at each iteration. The process terminates after $O(\log L)$ steps.

For the general case, each node can exchange information with more than a pair of nodes. This is because a node can be on more than one chain of nodes. The exact number of nodes involved can be as high as the degree of the node in the model graph. Additional information has to be exchanged to indicate which left or right substrings should be updated since a node can exchange information with another node multiple times. Moreover, the uniqueness and two-neighbor constraints are continuously applied to the new left, center and right substrings. Substrings that do not satisfy this constraint are rejected.

As an illustration of the more general case, consider the example shown in Figure 1(b). The initial left and right substrings obtained for the various matching positions are shown in Table 1 (and Table 2 for Figure 1(c)). The center substring (not shown in the third column of the Table) is the node itself. Node a (same for node d) matches position 1 (and 3). For that position, one of the initial substrings are (b, a, e).

Nodes	Positions	Substrings $(left, right)$
a, d	1	$(b,e),(c,e),(f,e)$
	3	$(e,b),(e,c),(e,f)$
b	0	$(a,d),(d,a)$
c, f	0	$(a,d),(a,e),(d,a),(d,e),(e,a),(e,d)$
e	1	$(c,a),(c,d),(f,a),(f,d)$
	2	$(a,d),(d,a)$
	3	$(a,c),(a,f),(d,c),(d,f)$

Table 1: Initial positions and substrings for an end view.

Consider what happens in node b at the first iteration. It exchanges information with nodes a and d. From node a it is supposed to insert the left substring e with a at position 3 (i.e. with e to be placed at position 2) and the right substring e at position 2 with a at position 1. This produces the substrings (ea, b, d) and (d, b, ae) with node b at position 0. With the information obtained from node d, the result is the same. Note that redundant substrings are removed. Since $L = 4$, the iteration stops. It turns out that the node strings generated—*bdea* and *baed*—match the contour string. These are also the node strings for nodes b, d and e.

A more interesting example is node c (or f). This node communicates with three other nodes: a, d, and e. Using information from a, it gets the substrings: (d, c, ae), (ea, c, d), (ea, c, e), (e, c, ae). The first two substrings are eliminated by the two-neighbor constraint since the nodes a, d, and e are all neighbors of c. The last two substrings are eliminated by the uniqueness constraint. Similarly the substrings (a, c, de), (ed, c, a), (ed, c, e), (e, c, de) obtained through communication with node d are eliminated. Communication with node e produces the substrings (a, c, ea), (d, c, ea), (ae, c, a), (ae, c, d), (a, c, ed), (d, c, ed), (de, c, a), and (de, c, d) all of which are eliminated by the uniqueness and two-neighbor constraints. Thus node c (and f) cannot be in the node string.

Nodes	Positions	Substrings $(left, right)$
a, d	0	$(b,c),(b,f),(c,b),(c,f),(f,b),(f,c)$
b	1	$(a,c),(a,f),(d,c),(d,f)$
	2	$(c,f),(f,c)$
	3	$(c,a),(c,d),(f,a),(f,d)$
c, f	1	$(a,b),(d,b),(e,b)$
	3	$(b,a),(b,d),(b,e)$
e	0	$(c,f),(f,c)$

Table 2: Initial positions and substrings for a top view.

For the top view example, the node strings are *ecbf* and *efbc*. The nodes which computed these strings are nodes b, c, e, and f. Nodes a and d produce no node strings.

After the node strings are found, they are sent to the host making sure that at only one processor per model graph communicates with the host. Each processor that has a node string checks to see if its own node number is the smallest in the node string. If so then it sends its node strings to a designated processor in its group (i.e. the processors that implement the model graph). This processor might be designated by simply having the lowest node number or processor address among the group of nodes or processors implementing the model graph. After collecting all these node strings, the designated processor sets a flag to indicate that it has at least one node string. This flag is then used to select the processors for accessing their node strings. Knowing the selected processors and their node strings, the host can determine which model and view match the contour string.

In estimating the time complexity of the algorithm, it is useful to note that the time it takes to exchange messages is the most important. If one assumes that this is to be true and that at any iteration, information can be exchanged with at most N nodes where N is the maximum number of nodes in a model graph, the algorithm takes $O(N \log L)$ information exchanges.

5 Conclusion

This paper shows how a parallel matcher for shape recognition in the rocks world can be implemented on the Connection Machine. The matcher uses constraints derived from qualitative models. For any given occluding contour the time it takes to find at one match is the same as that it takes to find any number (including zero) of matches. This means that it is relatively easy to determine if a new model

needs to be added to the data base. The set of matches computed form an equivalence class which can be used as a data base (a much smaller one) for a more elaborate and accurate matcher that uses more quantitative models. Without reducing the size of the data base such a matcher might turn out to be too expensive computationally to use.

6 Acknowledgment

The author would like to thank Rod Brooks for his comments in this work.

References

1 *The Essential *Lisp Manual.* Thinking Machines Corporation, Cambridge, Massachusetts, 1986. Thinking Machines Technical Report 86.15.

2 Dana H. Ballard and Christopher M. Brown. *Computer Vision.* Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1982.

3 W. Daniel Hillis. *The Connection Machine.* The MIT Press, Cambridge, Massachusetts, 1985.

4 W. Daniel Hillis and Guy L. Steele, Jr. Data parallel algorithms. *Communications of the ACM*, 29(12):1170–1183, December 1986.

5 Willie Y-P. Lim. *Fast Algorithms for Labeling Connected Components in 2-D Arrays.* Technical Report, Thinking Machines Corporation, Cambridge, Massachusetts, November 1986. To be published.

6 David Marr. *Vision.* W. H. Freeman and Company, San Francisco, California, 1982.

7 J. C. Wyllie. *The Complexity of Parallel Computations.* Technical Report TR 79-387, Department of Computer Science, Cornell University, Ithaca, New York, August 1979.

PARALLEL OPTICAL FLOW COMPUTATION

James Little, Heinrich Bulthoff and Tomaso Poggio

Massachusetts Institute of Technology
Artificial Intelligence Laboratory

Abstract

We outline a new, parallel and fast algorithm for computing the optical flow. The algorithm is suggested by a regularization method that we call "constraint method". The method, based on a theorem of Tikhonov, enforces local constraints and leads to efficient, parallel algorithms. The specific constraint exploited by our algorithm can be shown to correspond, in its most general form, to 3-D rigid motion of planar surfaces. An initial segmentation of the motion field can be obtained from the optical flow field generated by the algorithm. We also suggest an iterative scheme that can provide fast, approximate solutions and refines them subsequently. We discuss the implementation of the algorithm on the Connection Machine^{TM} system and its near real-time performance on synthetic and natural images.

1. Introduction

The computation of motion is an important module of early vision. It is potentially useful for computing the 3-D structure of surfaces, for segmenting a scene into objects and as a control module for navigation. The 3-D motion field, however, or even its 2-D projection, cannot be directly measured from the time sequence of images. Much effort in recent years has been invested in analyzing ways to compute the optical flow – the 2-D field of motion of brightness in the image. A common assumption is that the optical flow, suitably defined, is a very close approximation to the 2-D projection of the true 3-D motion field. Verri and Poggio (these Proceedings) argue against this assumption and the related use of the optical flow to obtain a precise quantitative estimate of 3-D structure and motion under general conditions. They argue also, however, that qualitative properties of the optical flow are very useful for segmenting the scene into different objects and for providing information about the *type* of motion and the 3-D structure, and that they are more robust than quantitative estimates. Notice that a closed-loop control system does not need accurate estimates. Furthermore, Verri's analysis suggests that the precise definition of optical flow is not critical as long as it preserves the qualitative properties of the 2-D motion field.

In this paper we outline a new, robust and fast algorithm for computing a version of the optical flow that was suggested by a regularization method that we call "constraint method". The algorithm has been implemented on the Connection Machine^{TM} system. Comparatively little work in the area of motion computation has made use of sequences of real images, because of technical limitations in acquiring and processing them. Our algorithm, which runs in times that are typically of the order of a few seconds, is routinely used with image sequences grabbed by the Vision machine's eye-head system (Poggio and staff, these Proceedings).

2. The algorithm

We can formulate a first algorithm in terms of the simplest possible assumption, that the optical flow is locally uniform. This assumption is strictly only true for translational motion of 3–D planar surface patches parallel to the image plane. It is a restrictive assumption that, however, may be a satisfactory *local* approximation in many cases. Let $E_t(x,y)$ and $E_{t+\Delta t}(x,y)$ represent transformations of two discrete images separated by time interval Δt, such as filtered images or a map of the intensity changes in the two images (more generally, they can be maps containing a feature vector at each location x,y in the image). We look for a motion displacement (also discrete) $\underline{v} = (v_x, v_y)$ at each discrete location x,y such that

$$\|E_t(x,y) - E_{t+\Delta t}(x+v_x\Delta t, y+v_y\Delta t)\|_{\text{patch}_{x,y}} = \min \quad (1)$$

where the norm is over a local neighborhood centered at each location x,y and $\underline{v}(x,y)$ is assumed constant in

the neighborhood. Equation (1) implies that we should look at each x, y for $\underline{v} = (v_x, v_y)$ such that

$$\int_{\text{Patch}_i} |\tilde{E}_t(x,y)\tilde{E}_{t+\Delta t}(x + v_x\Delta t, y + v_y\Delta t)| dxdy \quad (2)$$

is maximized. Equation (2) represents the correlation between a patch in the first image centered around the location x, y and a patch in the second image centered around the location $x + v_x\Delta t, y + v_y\Delta t$.

This algorithm can be translated easily into the following description. Consider a network of processors representing the result of the integrand in equation (2). Assume for simplicity that this result is either 0 or 1. (This is the case if E_t and $E_{t+\Delta t}$ are binary feature maps.). The processors hold the result of multiplying (or logically "anding") the right and left image map for different values of x, y and v_x, v_y. The next stage, corresponding exactly to the integral operation over the patch, is for each processor to count how many processors are active in an x, y neighborhood at the same disparity. Each processor thus collects a vote indicating support that a patch of surface exists at that displacement. The last stage is to choose $\underline{v}(x,y)$ out of a finite set of allowed values that maximizes the integral. This is done by an operation of "non–maximum suppression" across velocities out of the finite allowed set: at the given x, y, the processor is found that has the maximum vote. The corresponding $\underline{v}(x,y)$ is the velocity of the surface patch found by the algorithm. This algorithm is similar to the stereo algorithm implemented by M. Drumheller and T. Poggio (1986) on the Connection Machine^{TM} system.

The algorithm will not find, in general, the 2-D projection of the true 3-D velocity field. This will happen only when the features used for matching correspond to markings on the 3-D surfaces and when either the features are sparse (no ambiguity) or the disambiguation step (the voting and non-maximum suppression stage) finds the true correspondence (i.e. the underlying assumptions are satisfied). Even when the result is not the true motion field, the algorithm will usually preserve its most important qualitative properties.

3. The constraint method

The algorithm just described was suggested by a regularization method (Poggio and Verri, in preparation) that follows directly from some results of Tikhonov and the discrete nature of the image data. We outline here the basis of the constraint method and then discuss how it is implemented by our motion algorithm. We postpone a more formal discussion to a later paper.

3.1. Tikhonov Lemma

As pointed out by Poggio and Torre (1984) many problems of early vision are ill-posed. For example solutions to problems such as surface reconstruction, computation of visual motion and depth from stereo are not unique or they are not stable (for instance for edge detection and structure from motion). A lemma by Tikhonov and Arsenin (1977) suggests how to regularize these problems by exploiting the discrete nature of the image data and the boundedness of sought solutions (for instance depth of surfaces or velocity values).

Let us consider the problem of solving the equation

$$A\underline{x} = \underline{y} \quad (3)$$

for $\underline{x} \in X$, X a metric space. Let $\underline{y} \in Y$, Y a metric space, and let A be an operator mapping $D(A) \subseteq X$ onto $R(A) \subseteq Y$. In many applications it is required that the solution $\underline{x}$ to (2.1) i) exists, ii) it is unique and iii) it depends continuously on $\underline{y}$. A problem whose solutions satisfies i), ii) and iii) is said to be *well-posed*; otherwise it is said to be *ill-posed*. It is clear that the solution does not exist if $\underline{y} \notin R(A)$ and that it is not unique if A is not injective. The solution depends continuously on $\underline{y}$ when the inverse of A, A^{-1}, is continuous. Let us assume that the solution to (2.1) is given by

$$\underline{x}_0 = \inf_{x \in X} \|A\underline{x} - \underline{y}\| \quad (4)$$

Sufficient conditions for the well-posedness of (3) (or (4)) are provided by the following lemma by Tikhonov:

Lemma Suppose that the operator A maps a compact set $F \subseteq X$ onto the set $U \subseteq Y$. If $A : F \rightarrow U$ is continuous and one-to-one, then the inverse mapping $A|_U^{-1}$ is also continuous.

The uniqueness is obviously guaranteed by the fact that A is one-to-one while both the existence and the stability of the solution rely upon the compactness of the set F. Let us assume that X (or F) is a closed and bounded subset of R^n. Therefore X (or F) is compact. Hence, if the solution belongs to a closed and bounded set in R^n and A is one-to-one, the problem to solve is well-posed. Since every early vision problem is defined on a n-dimensional space (where n is the number of pixels), sufficient conditions for the well-posedness of these problems can be given setting *a priori* bounds on the solutions. In the specific case of our motion algorithm the compactness assumption is satisfied by restricting the input and output spaces to be discrete and finite (only a small set of discrete velocities is allowed).

It is important to stress that in several early vision problems the solution is not unique. For example the same 2-D motion field can be generated by the projection of different kinds of 3-D motion fields. This corresponds to the problem of solving equation 2 when A is not one-to-one. In this case the lemma is not sufficient since it does not guarantee uniqueness. Let us assume, however, that the solution is still bounded: only a finite number of solutions, then, are possible due to the quantization. Uniqueness can be achieved by giving a set of *rules* that allow the selection of at most one value (possibly none) for the solution at every point: such a strategy obviously will always succeed due to the finite number of possible values at each location. It is worth noting that the solution could actually turn out to be defined only in some locations, corresponding to the selection of no value. The set of rules corresponds in our algorithm to the voting followed by non-maximum suppression.

3.2. Equivalent physical constraints

The algorithm can be extended to a less restrictive constraint: that the optical flow is locally linear or even quadratic instead of simply constant. The quadratic case corresponds to a very simple constraint on 3-D motion and surfaces, *under the assumption* that the optical flow is sufficiently close to the 2-D motion field. Under this assumption we can use the following result due to Waxman (1986): *the velocity field on the image plane originated by arbitrary, rigid 3-D motion of a planar surface patch is quadratic.* In this way, the algorithm exploits the physical assumption that surfaces are—at least relative to the image resolution—locally planar (the "allowed" world is thus a world of polyhedral solids, albeit with a very high number of faces). It is worth noting that our initial experiments indicate that the quadratic and even the linear assumptions do not change significantly the results obtained with the "constant constraint" algorithm.

3.3. An iteration scheme

The implementation of the quadratic patch constraint is computationally expensive, even for coarse discretization of the velocity values. Though it may be unnecessary to consider in practice quadratic patches, it is of interest to develop a scheme that allows for a fast approximate solution based on the constant field assumption that is then refined in terms of higher order assumptions such as linear and quadratic patch. It is natural to consider an iteration that first finds the best "constant" solution, then refines it with the best "linear" correction and finally finds the best "quadratic" correction. In general, the best quadratic correction does not provide the best quadratic approximation. Results however about the estimation of polynomial operators (see for instance Poggio, 1975, theorem 4.2) suggest that iterating the procedure should converge to the best quadratic approximation. In this way we can find the best "constant" estimation of the optical flow and then refine it by successive iterations that cycle from the lowest to the highest order and to the lowest again.

4. The Connection Machine^{TM} implementation

The time Δt between images is small, on the order of one video time frame (1/30th second). During this short time, the appearance of a moving object can change due to its own motion, camera motion, light source motion, or all three, among other effects (see Verri and Poggio, this Proceedings). However, when the local intensity variation in the surface albedo is sufficiently large, the errors introduced by these effects are minimized. For this reason, we use the output of an edge detection step as the input to later stages of optical flow computation. Both the Laplacian of a Gaussian and Canny's edge detector (Canny, *1986*) have been used with good results. Let E_t be an image containing a description of the features at time t. Features can, for example, describe the sign of the x and y components of the image gradient at edge points.

The comparison of features in E_t and $E_{t+\Delta t}$ is performed for each $\underline{v}$ and is recorded at each x, y in a matching map $M(\underline{v})$, which identifies whether the feature at $E_t(x, y)$ found a match in $E_{t+\Delta t}(x, y)$ when displaced by $(v_x, v_y)\Delta t$ in the image plane. This process is spatially parallel, operating at all x, y simultaneously, for each $\underline{v}$ in the discrete set that is allowed. The process iterates over this range, generating the matching map $M(\underline{v})$.

The integration stage acts only upon the matching maps $M(\underline{v})$. By assumption, we are looking for the optical flow which is locally constant. We choose the flow $\underline{v}$ which, in a neighborhood N around x, y, maximizes the number of matches. Again this procedure iterates over all $\underline{v}$ in the bounded range. We identify a displacement only at those locations x, y at which the maximum vote is unique; ties are ambiguous and are eliminated. The result is a map of the optical flow at locations in the image at which a moving feature has undergone unambiguous motion.

The comparison and integrated stages could be merged into one stage, in which case we would be directly implementing a binary correlation scheme on the feature maps.

The constraint method can also be used to exploit more sophisticated and less restrictive assumptions about surfaces. We could, for instance, assume that the motion surface is locally planar, or that the patch is locally quadratic. These more general assumptions can be implemented by changing appropriately the geometry of the neighborhoods over which the voting takes place. We have implemented the iterative scheme for determining the best linear correction (linear in x, y) to the constant solution. There, the separation of the two stages is crucial for a fast implementation; for corrections that are not constant, the voting neighborhood no longer corresponds to a simple patch in the matching map of just one displacement. Also, the comparison operation can be done once and used for all later integration stages.

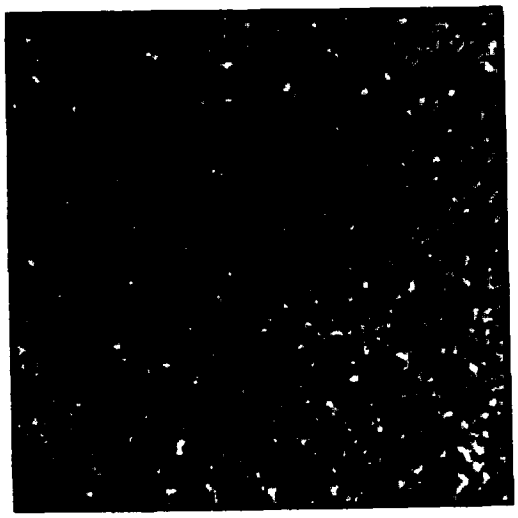


Figure 1 A disc with random texture is embedded in a background pattern with the same texture. The shape of the object becomes immediately visible when foreground and background pattern are moving with different speeds (Figure 2).

4.1. Examples

We used both synthetic and real images for testing the implementation of the algorithm on the Connection Machine^{TM} system. The algorithm requires as much structure as possible in the images to be analyzed. In the synthetic images we produce sufficient structure by using random textures for the foreground and background patterns [1] (Figure 1). Since the foreground has the same texture as the background it remains invisible to the human observer as long as it is not moving. As soon as either the foreground or the background pattern begins to move the object becomes immediately visible. The needle diagrams of the opti-

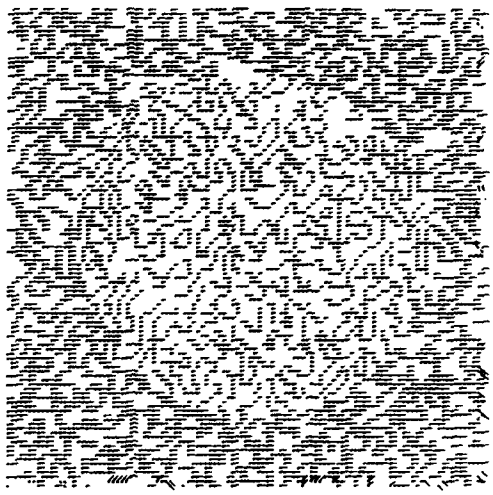


Figure 2. Needle diagram of the motion field computed by the constraint method. A disc with random texture is moving with half speed in the same direction as the background. The algorithm computes consistent velocities for foreground and background motion.

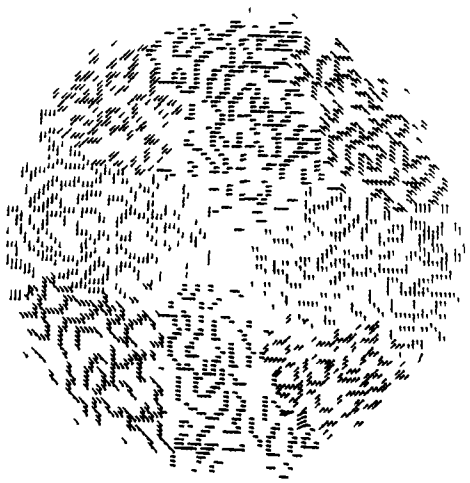


Figure 3. Needle diagram of the motion field computed by the constraint method. A disc with random texture is rotating clockwise at 2 degrees per time step in front of a stationary background.

[1] To prevent antialiasing in the motion computation output the images were appropriately bandpass filtered

cal flow field in Figure 2 show that the algorithm succesfully computes consistent motion if foreground and background patterns move with different speeds. In Figure 3 the disc-like foreground pattern is rotating clockwise in front of a stationary background. Again all motion vectors show locally consistent direction and magnitude of velocity for the foreground pattern. Figure 4 shows that the Connection Machine^{TM} implementation of the constraint method is able to compute a consistent description of motion in a natural sequence of images. Two images are digitized in 30 msec intervals with the Vision machine's eye-head sytem and analyzed by the Connection Machine^{TM}. Almost all of the significant intensity changes on the moving robot are labelled with motion vectors pointing to the left which is consistent with the actual motion of the robot. The non-moving chair and the background is invisible to the motion detection mechanism. Only a few small patches show false motion due to apparent motion of specular reflections caused by changes in illumination between frames. On a Connection Machine^{TM} having 16K processors, the optical flow of a 128 × 128 image can be computed in a few seconds, several hundreds of times faster than on a Symbolics 3640 Lisp Machine.

5. Finding discontinuities

We are presently analyzing several methods for finding efficiently initial estimates of the discontinuities of the optical flow that may be later refined (for instance at the integration stage). We give here a very brief outline of three methods:

5.1. Statistics of the voting step

At motion discontinuities the assumption of a constant (or linear or quadratic) motion field is obviously wrong. One would expect therefore that the "votes" at a motion discontinuity (say in the case of the "constant" motion algorithm) would fail to support clearly any single velocity. In fact, regions of close "ties", or equivalently of winners with locally minimum votes, often delinate motion discontinuities. Our implementation of this procedure scales the number of votes at a location by the total number of features in the voting neighborhood. Close ties receive values near 0.5. Figure 5 shows the result of thresholding this ratio at 0.75. This idea can be developed further by considering more complete statistics of the votes (Spoerri and Ullman, in preparation).

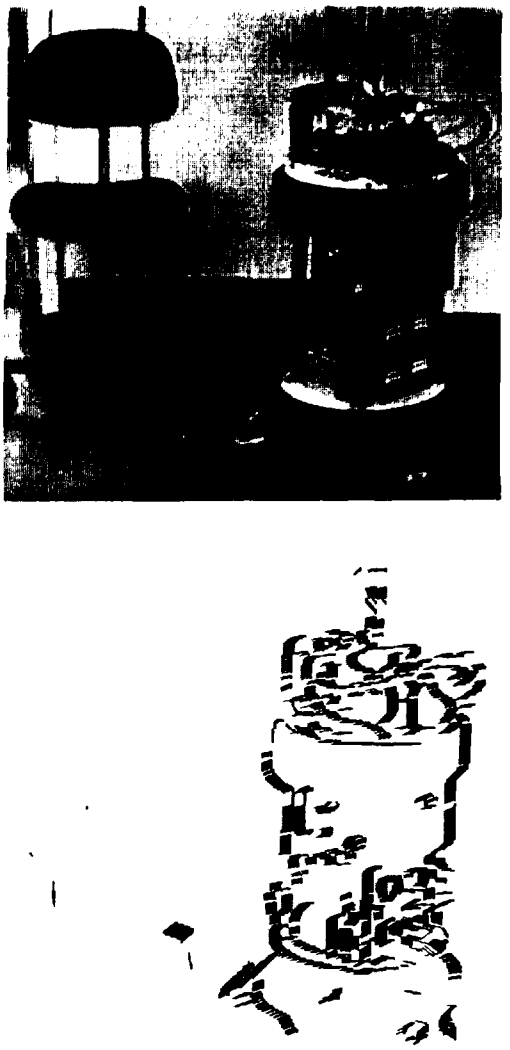


Figure 4a. Frame 1 of 2 from a motion sequence used to test parallel motion detection algorithms with real images (256 × 256). The robot is moving to the left. b. Output of the constraint method algorithm for the motion sequence shown above. Most of the needles point to the left consistently with the actual movement. The stationary background (wall, floor and chair) is invisible to the motion detectors. The small patches of indicated motion on the left are most probably caused by apparent motion of specular reflections due to changes in illumination between frames.

5.2. Vetoing coherent motion

Motion discontinuities can be found by using an algorithm that was suggested by data on the insect visual system (Reichardt et al., 1983). The idea is to inhibit or veto the value of the optical flow at each point by the average value of the field over a large region centered at that point *whenever* the motion is of the same

type. The scheme suggested by the insect work is the following: at each x, y consider separately the x and the y component of the optical flow, take its value and divide it by the average value, computed over a large region. Figure 6 shows the output of this operation on two examples. The average may be Gaussian weighted. It is quite intriguing to notice that the basic operation is very similar to a recent proposal by Land (1986). It is also similar to performing a center-surround operation (such as the Laplacian of a Gaussian, but with much larger surround) on the log of the optical flow.

5.3. Edge detection on the optical flow

Edge detection on the each component of the optical flow is the simplest way to obtain an initial estimate of discontinuities. We plan to use Canny's edge detector (Canny, 1986) on each component of the optical flow.

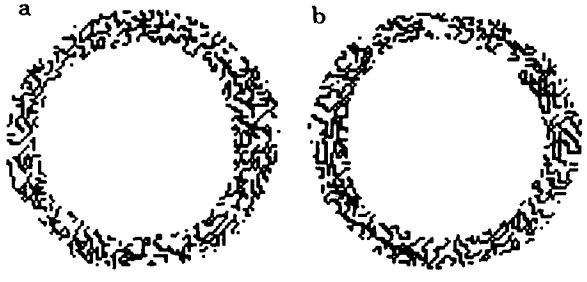


Figure 5. Edge labeling by relative motion. Motion discontinuities can be found by finding the locations that get a minimum number of votes for consistent motion. This should be the case at object boundaries for relative motion between foreground (object) and background. a. A disc moves with same speed but in opposite direction to the moving background. b. Object moves in same direction but half the speed of the background.

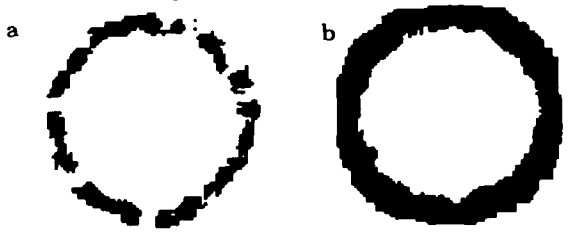


Figure 6. Edge labeling by relative motion. a. A disc with random texture is moving with opposite direction in front of a moving background with the same texture. Only those indicators of relative motion with values above a certain threshold are shown here. b. The same pattern is moving in front of the same background in the same direction but with half the velocity. The size of the labeled area depends on the size of the larger mask (see text).

Reading list

Canny, J.F. (1986) A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence* Nov. 1986, Vol. 8, No. 6, 679-698.

Drumheller, M. and Poggio, T. (1986) Parallel Stereo *Proceedings of IEEE Conference on Robotics and Automation*, San Francisco.

Hassenstein, B. and Reichardt, W. (1956) Systemtheoretische Analyse der Zeit-, Reihenfolgen- und Vorzeichenauswertung bei der Bewegungsperzeption des Rüsselkäfers *Chlorophanus*. *Z. Naturforsch.* **IIb**, 513–524.

Hildreth, E. C. (1984a) *The Measurement of Visual Motion*. Cambridge: MIT Press.

Horn, B. K. P. and Schunck, B. G. (1981) Determining optical flow. *Artif. Intell.* **17**, 185–203.

Land, E.H. (1986) An alternative technique for the computation of the designator in the retinex theory of color vision. *P.N.A.S.* **83**, 3078-3080.

Marr, D. and Ullman, S. (1981) Directional selectivity and its use in early visual processing. *Proc. R. Soc. London Ser. B* **211**, 151–180.

Poggio, T. (1975) On optimal nonlinear associative recall. *Biol. Cybern.* **19**, 201–209.

Poggio, T. and Torre, V. (1984) Ill-posed problems and regularization analysis in early vision. Massachusetts Institute of Technology Artificial Intelligence Laboratory Memo 773, C.B.I.P. Paper 001.

Poggio, T. and Reichardt, W. (1973) Considerations on models of movement detection. *Kybernetik* **13**, 223–227.

Reichardt, W., Poggio, T. and Hausen, K. (1983) Figure–ground discrimination by relative movement in the visual system of the fly. Part II: Towards the neural circuitry. *Biol. Cybern.* **46**, 1–30.

Tikhonov, A.N. and Arsenin, V.Y. (1977) *Solutions of ill-posed problems*. W.H.Winston, Washington, D.C.

Torre, V. and Poggio, T. (1978) A Synaptic mechanism possibly underlying directional selectivity to motion. *Proc. R. Soc. Lond. B* **202**, 409–416.

Ullman, S. (1981) Analysis of visual motion by biological and computer systems. *IEEE Computer*, August 1981, pp. 57–69.

Waxman, A. M. (1986) Image flow theory: A framework for 3-D inference from time-varying imagery. In *Advances in Computer Vision*, ed. C. Brown, New Jersey: Erlbaum. *In press.*

Using Optimal Algorithms to Test Model Assumptions in Computer Vision

Terrance E. Boult*
Columbia University Computer Science Department

Abstract

To develop useable alrotithms to solve problems in computer vision one inevitably has to make a number of assumptions about the problem, and about the world. At times, we are able to derive optimal error algorithms. Unfortunately, as many of us have seen first hand, "optimal" algorithms, when run on *real* data, can give very poor results. As a result, the developed algorithm is then laid to rest, a death caused by assuming a world model which does not coincide with reality. However, such optimal algorithms may still serve a purpose in computer vision. At times, these algorithms may be used to help test the assumptions which we can make about the world and thus develop better models of the world or beter models of animal visual system.

After a general discussion of the use of optimal algorithms in verification of model assumptions, we present an example from the modeling of human reconstruction of surfaces from sparse visual depth data. We then discuss the interplay of contaminated data and modeling. We end with a short discussion of the interplay between optimal error algorithms, algorithm complexity, and modeling.

1 Introduction

A large part of the research in computer vision can be viewed as development of models for the extraction of information from the visual data or development of models of various components in animal visual systems. Because most vision problems are ill-posed, the models developed must contain various assumptions about the world. Unfortunately, analytical analysis is not practical for models of all processes. Thus many researchers have turned to computational analysis (simulation) to provide them with means of comparing or validating models. In addition to the traditional problems of assessing the difference between experimental behavior and model predictions caused by experimental errors and factors not considered in the model, the researcher using computational analysis must deal with errors which are caused by the inaccuracy of the "computational embodiment of the model".

We highlight this last difficulty by a simple example. Suppose based on analytical methods we can predict one of two models, M_1 and M_2, both of which are based on the minimization of a distance functional. For the sake of argument let us assume the distance functional is non-convex for M_1 but is convex for M_2. Furthermore, let us assume that M_1 is the correct model. As often happens in practice, we cannot test the two models directly, but rather we implement algorithms (say φ_1 and φ_2) which solve example problems (assuming models M_1 and M_2 respectively) and compare the results of a number of computational experiments. For simplicity, let us assume the "quality" of a computational experiment is proportional to the difference between the a known distance and the estimated (computed) minimal distance proposed by the model. Further, let us assume that both algorithms use a simple gradient descent algorithm to find the minimum of their respective distance functionals. After a number of computational experiments, we would most likely conclude that model M_2 was better than model M_1.[1] This incorrect conclusion is not caused solely by the differences in the models but by the error properties of the algorithms used in the computational analysis. While the "computational error" was obvious in this case (using gradient descent for a non-convex minimization problem), in practice subtle differences in algorithms (and assumptions about the problem's definition) can also cause large differences in the computational error.

The above example shows that the comparison of algorithms does not necessarily imply anything about the underlying models. In the next section we discuss how this changes if the algorithms are optimal error algorithms, and how this can be used to verify model assumptions or choose between different models. Then in Section 3 we present an example of the general approach applied to problem of visual surface reconstruction. We end this paper with a discussion of the interaction of optimal error algorithms and complexity.

2 General Approach

There are four phases in the general approach of using optimal error algorithms to choose between different mathematical models of a process. These are:

1. the precise formulation of different models of the process
2. the theoretical derivation of the optimal error algorithm[2]

*This work supported in part by DARPA grant # N00039-84-C-1065 and NSF grant# DCR-82-14322.

[1] This conclusion assumes that the gradient descent algorithm would often get stuck in local minima far from the global minimum of the non-convex distance function, which is generally the case when applying it to non-convex functions.

[2] By optimal error algorithm we mean the algorithm with minimal error. We therefore assume that the user has chosen a measure of error, e.g. maximum over all possible problem instances of the worst case difference between the true solution and the approximation. The results of the process will be more significant if the algorithm is strongly optimal, i.e. it has minimal error for every set of initial data.

for each model given in phase 1,

3. the implementation (and analysis of the numerical stability) of the optimal error algorithms derived in phase 2

4. the comparison of the different models, based on the interpretation of the computational results of the algorithms in phase 3. This is generally accomplished using a finite but representative sample of problem elements.

For the approach to be applicable, each phase must be completed. The importance of phase 1 should be obvious. To see the importance of phase 2, note that if we consider two optimal error algorithms (say φ_1 and φ_2) for models M_1 and M_2, then we know that the computational error for each algorithm is inherent in the problem. If we find φ_1 to be better than φ_2, then we can draw one of three conclusions:

- M_1 is a better model than M_2.
- the error inherent in any computational embodiment of M_2 is so large that M_1 is effectively better since the error is inherent in the model assumptions. No process, including the one being modeled, could compute better approximations.
- M_2 is a better model that M_1, but our theoretical measure of error and our sampling of data for the comparison are such that the error of M_1 on that sample is considerably smaller than its theoretical error. (Therefore the "representative" samples were not representative of the errors inherent in the model.)

Thus if samples in phase 4 are truly representative, the problem is well conditioned and phase 3 shows that the implementation of the algorithm is well conditioned and numerically stable, then the comparisons in phase 4 result in a comparisons of the models, not just the algorithms.[3]

We note that any one of the phases may, and often will, be difficult if not impossible. Obviously phase one and phase four should be in the domain of knowledge of the scientist doing the modeling. Phase 2, which may be the most difficult for a scientist outside of computer science, may often be accomplished by referring to previous work in optimal algorithms, see [Traub-Woźniakowski-80], [Traub-Wasilkowski-Woźniakowski-83], and [Micchelli-Rivlin-77]. Finally phase three may be accomplished by consulting a numerical analyst. While the four phases might call for considerably more effort than a simple comparison of different algorithms which solve a problem assuming different models, the extra effort turns a simple comparison of different algorithms into a experiment which compares different models of a process.

3 Example: Reconstruction of Surfaces from Sparse Depth Data

In this section we discuss some of the details of the use of optimal error algorithms to study models for the process of surface reconstruction in the human visual system.

[3] A problem is well-conditioned if small changes in the input do not result in wild variations in the solution. An algorithm is numerically stable if the computed result is the exact solution to a slightly perturbed problem.

Since the pioneering work of Julesz, (see [Julesz-71] and the many references therein), it has been known that when presented with sparse depth data the human visual system "perceives" a dense surface rather than a collection of unconnected points in depth.[4] The perception is both vivid and stable. Furthermore, it is the perception of "smooth" three dimensional surfaces. There has been considerable psychological research investigating the phenomenological aspects of the process. In recent years there have been a number of computational models of how this process might work. These computational models are of great interest to the computer vision community.

Researchers at MIT, see [Grimson-79], and [Grimson-81], proposed that we formulate the problem (assuming no noise in the depth data) as one of finding a surface passing through the data that minimizes a given measure of surface unreasonableness.[5] In their work, they assumed the world contained surfaces whose second derivatives are in L^2 and that the measure of reasonableness was given by the bending energy of a thin-plate. That is the surfaces are such that if one takes the the second derivative of the surface, and integrates the square of these values over the surface, the integral is bounded. The bending energy of a thin-plate is given by the second Sobolev semi-norm, i.e.

$$\left\{ \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \left(\left(\frac{\partial^2 f}{\partial x^2} \right)^2 + 2 \cdot \left(\frac{\partial^2 f}{\partial x \partial y} \right)^2 + \left(\frac{\partial^2 f}{\partial y^2} \right)^2 \right) dx\, dy \right\}$$

While Grimson presented various arguments why to use this measure, there was no formal attempt to show this model was appropriate for human vision, nor that it was the most appropriate for computer vision.

In [Boult-86], we initiated the application of optimal error algorithms as a tool for modeling, by applying the general approach of Section 2 to this problem. We generalized the model proposed by Grimson by allowing other classes of functions and different measures of unreasonableness of a surface.[6] For each model we implemented an optimal error algorithm and then, using psychological experimentation, compared the models.

3.1 Experimental Procedure

There were 9 different data-sets considered, each generated by a different underlying surface. These were considered to be representative of the surfaces we would encounter in computer vision. We describe each of these underlying surfaces in turn and the way in which that data was sampled.

Data-set #1 A basis function using randomly located data. The data at all information points has value 0 save one which has value 1.

Data-set #2 A small "wedding cake" produced by stacking up three planes of height 0, $\frac{1}{2}$, and 1 and sampling on a 10 by 10 grid.

[4] Sparse depth data is presented by means of random-dot stereograms or dicoptic displays

[5] This minimization property was proposed in part because the minimization of an energy function is an operation that might be performed by a neural network, see [Grimson-81]. The later work in [Terzopoulos-84] also makes these assumptions.

[6] These correspond to different assumptions about "smooth " surfaces in the world. Because of the arguments about neural-network type minimizations, we maintained that aspect of the model.The strongly optimal error properties of the algorithms follow from theoretical work in a very general setting, see [Lee-85] or [Boult-86]. It is interesting to note that the algorithms proposed by the MIT researchers are *discrete approximations* to the optimal error algorithm for the class they chose.

Data-set #3 100 randomly located points taken from a quarter of a wedding cake containing four planes with heights 0, $\frac{1}{3}$, $\frac{2}{3}$, and 1.

Data-set #4 A parabolic sheet (height 0 to .9) defined by only 16 regularly spaced points.

Data-set #5 A half-cylinder of radius $\frac{1}{2}$ lying on a plane, sampled at 100 randomly located points in the unit square.

Data-set #6 A quarter of a wedding cake (heights of the four planes are 0, $\frac{1}{3}$, $\frac{2}{3}$, and 1) defined on a 10 by 10 regular grid.

Data-set #7 A noisy approximation to the central portion (with rectangular boundary) of a hemi-spherical surface sampled at 100 randomly placed points.

Data-set #8 A surface defined by two planes meeting at an acute angle then a parabolic sheet meeting the right edge of one of the planes. Data sampled on a regular 10 by 10 grid.

Data-set #9 A saddle surface (i.e. rectangular hyperbolic sheet). In this case the data is located at 100 randomly located points. The range for the data was [-.9,.9].

From the initial data we reconstructed strongly optimal error estimates using a reproducing kernel spline based algorithm (see Boult-86]) under different formulation assumptions. The formuations (i.e. class/norm pairs) and their letters which will be used n subsequent references to them are:

$\mathcal{A}$: $D^{-2}H^{-.75}$ with the second Sobolev semi-norm,

$\mathcal{B}$: $D^{-3}H^{-1.75}$ with the third Sobolev semi-norm,

$\mathcal{C}$: $D^{-4}H^{-2.75}$ with the fourth Sobolev semi-norm.

$\mathcal{D}$: $D^{-2}H^{-.5}$ with the second Sobolev semi-norm,

$\mathcal{E}$: $D^{-2}H^{-.25}$ with the second Sobolev semi-norm,

$\mathcal{F}$: $D^{-2}L^2$ with the second Sobolev semi-norm,

$\mathcal{G}$: $D^{-2}H^{.5}$ with the second Sobolev semi-norm,

$\mathcal{H}$: $D^{-4}L^2$ with the fourth Sobolev semi-norm,

$\mathcal{I}$: $D^{-5}L^2$ with the fifth Sobolev semi-norm.

The definition of these classes and their associated semi-norms ppears else where (see [Boult-Kender-86] or [Duchon-76]). The eader can see sample reconstructions from classes $\mathcal{D}$ and $\mathcal{F}$ (the lass used by Grimson) for three data-sets in Figures 5—10.

In our psychology experiments to compare the models (phase of the general approach) the subjects were presented with 9 ackages each containing 4 views of the initial data and 36 samle reconstructions. The subjects rated each sample on "how ell it fit their impression of the surface generating the initial ata". They also compared each sample with 5 others stating if seemed a better, equal or worst fit to the initial data. Note his results in a tremendous amount of data. In total each of the ix subjects ranked 324 sample reconstructions and made 1485 airwise comparisons.

.2 Experimental Results

In the experiment there were two separate types of responses athered: direct rating and comparison. Since both types of esponses result in similar findings we present only the former. or more detail, including an individual analysis of every dataet, consult [Boult-86]. Because the subjective ranking of the lasses will depend on the underlying data, we shall break each nalysis up into groups:

- *Jump discontinuities* (data-sets 1, 2, 3, 6)
- Orientation discontinuities (data-sets 5, 8)
- *No discontinuities* (data-sets 4, 7, 9)
- Overall (data-sets 1, 2, 3, 4, 5, 6, 7, 8, 9).

Note that the performance in the overall category may reflect a basis in the selection of the initial data (i.e. how many surfaces with discontinuities were considered as compared to the number of smooth surfaces tested). For this reason one should not stress those results.

The analysis of the rating responses is simply the calculation of the median, mean, and variance of the rating of "quality". This data will be displayed for all class/norm pairs in graphical form. The graph will have nine columns, one for each class/norm pair. In each column the mean will be presented as a ×, the median presented as a o and the variance as a line of the appropriate length centered around the mean. In addition, above each column will appear the numerical value of $\overline{Quality}$, the mean quality response to three significant digits.

Throughout the discussion we shall consider the relationship between the mean rating responses and the "apparent number of derivatives" for each class. Note that many of the classes are defined such that the apparent number of discontinuities is a fractional value, see [Boult-86]. For the nine classes the apparent number of derivatives are: class $\mathcal{A}$–1.25, class $\mathcal{B}$–1.25, class $\mathcal{C}$–1.25, class $\mathcal{D}$–1.5, class $\mathcal{E}$–1.75, class $\mathcal{F}$–2.0, class $\mathcal{G}$–2.5, class $\mathcal{H}$–4.0, and class $\mathcal{I}$–5.0.

3.2.1 Analysis of Data-sets with Jump Discontinuities

The data on which this discussion is based was generated by combining the rating and pairwise comparisons from all subjects over those data-sets where the underlying surface has a jump discontinuity, i.e. data-sets #1, #2, #3 and #6. Examination of the rating responses, see Figure 1, suggests that classes $\mathcal{D}$, $\mathcal{A}$, $\mathcal{B}$ and $\mathcal{C}$ are best suited to this type of data, while classes $\mathcal{H}$, $\mathcal{I}$, and $\mathcal{G}$ are the poorest of those considered. If we consider the relationship between the apparent derivatives and the rating, we see that the peak in quality corresponds to an apparent derivative of 1.25, and that as we increase the number of apparent derivatives, the quality gradually deteriorates.

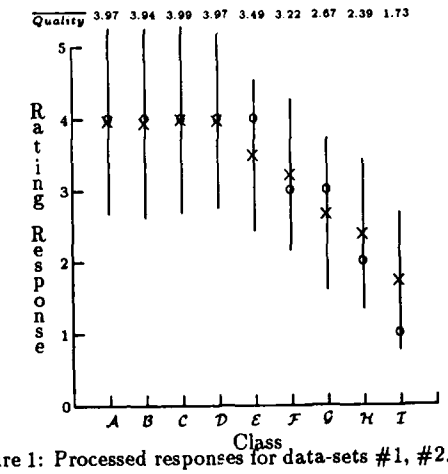


Figure 1: Processed responses for data-sets #1, #2, #3, and #6, i.e., the data-sets with jump discontinuities.

Let us now consider the quality of the reconstructions if we fixed the apparent differentiability of the class of functions in the

class and vary the norm. This can be done by considering classes $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{C}$ each of which has 1.25 apparent derivatives, but which minimize the second, third and fourth Sobolev semi-norm respectively. The rating responses (and comparison responses) for these three classes are virtually identical. Thus one might conclude that for these data-sets, the actual semi-norm minimized was not a dominant factor.

We can also consider the related question of fixing the norm and varying the class of functions. This is accomplished by comparing classes $\mathcal{A}$, $\mathcal{D}$, $\mathcal{E}$, $\mathcal{F}$, and $\mathcal{G}$. These classes all minimize the second Sobolev semi-norm but their functions are assumed to be in the semi-Hilbert spaces: $D^{-2}H^{-.75}$, $D^{-2}H^{-.5}$, $D^{-2}H^{-.25}$, $D^{-2}L^2$, $D^{-2}H^{.5}$ which have quality ratings of: 3.97, 3.97, 3.49, 3.22, and 2.67 respectively. Thus we see there is an inverse relationship between the apparent differentiability and the mean quality rating over data-sets with jump discontinuity.

3.3 Analysis of Data-sets with Orientation Discontinuities

The processed responses which appear in Figure 2 were generated by combining the raw responses from all subjects for data-sets #5 and #8. Examination of the rating responses in Figure 2 suggest that classes $\mathcal{F}$, $\mathcal{H}$, $\mathcal{E}$, and $\mathcal{G}$ are well suited to data whose underlying function has orientation discontinuities and that classes $\mathcal{I}$, $\mathcal{B}$, and $\mathcal{C}$ are poorly suited to such data.

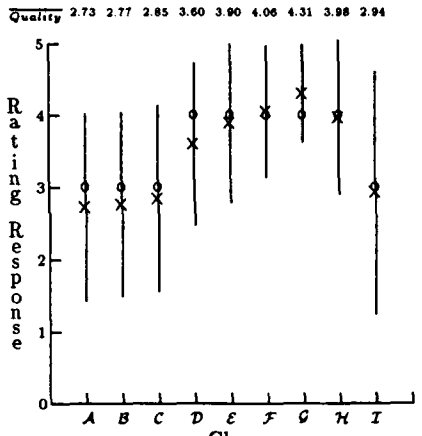


Figure 2: Processed responses for data-sets #5 and #8, i.e., those data-sets with orientation discontinuities.

Again let us now consider the quality of the reconstructions if we fixed the apparent differentiability of the class of functions in the class and vary the norm. The rating responses for classes $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{C}$ are virtually identical. Again, one might conclude that for these data-sets, the actual semi-norm minimized was not a dominant factor.

We can also consider the related question of fixing the norm and varying the class of functions. We see there is a unimodal (with peak at 2.5) relationship between the apparent differentiability and the mean quality rating over data-sets with orientation discontinuity.

3.4 Analysis of Data-sets with No Discontinuities

The responses presented in Figure 3 were generated by combining all subjects' responses over data-sets #4, #7, and #9. These data-sets were generated by underlying functions which were very smooth, in fact infinitely differentiable. Examination of the rating responses suggests that all classes except $\mathcal{D}$ and $\mathcal{A}$ are well suited to the problem, with classes $\mathcal{H}$ and $\mathcal{I}$ being slightly inferior to classes $\mathcal{F}$, $\mathcal{E}$, $\mathcal{G}$, $\mathcal{B}$ and $\mathcal{C}$. The comparison responses support the idea that classes $\mathcal{F}$, $\mathcal{G}$, $\mathcal{B}$ and $\mathcal{C}$ are best suited to the smooth data-sets.

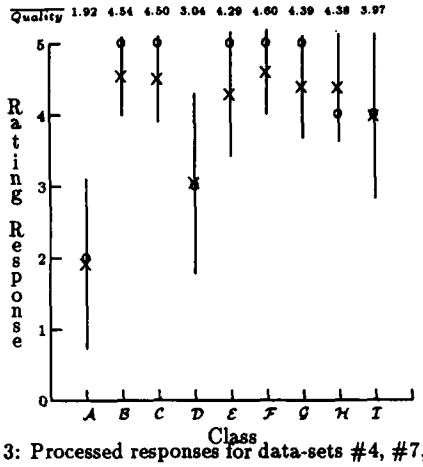


Figure 3: Processed responses for data-sets #4, #7, and #9, i.e., those data-sets with very smooth underlying functions.

Considering the relationship between norm and rating responses we see that classes $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{C}$ have quite different responses. This difference is due to the fact that reconstruction for class $\mathcal{B}$ and $\mathcal{C}$ are exact for polynomials of order 2 and 3 respectively. Thus they exactly reconstruct the underlying data for data-sets #4 and #9 (but not #7 because of the noise).

With respect to the relationship between apparent derivatives and responses (taking into account the fact that classes $\mathcal{H}$, $\mathcal{I}$, $\mathcal{B}$ and $\mathcal{C}$ were exact because of their null-spaces). We see there is a unimodal tendency (with peak around 2). However, because we exclude almost half the classes the result is considerably weaker than those for data-sets with jump or orientation discontinuities.

3.5 Analysis of Performance over All Data-sets

Examination of the overall ratings suggests that classes $\mathcal{F}$, $\mathcal{E}$, $\mathcal{B}$ and $\mathcal{C}$ are the overall most appropriate classes of those tested. We point out that this overall score is very dependent on the different individual data-sets, as is apparent from the large variance in rating responses.

3.6 Experimental Conclusion

We have demonstrated models which yield reasonable and sometimes far better solutions to the visual surface reconstruction problem than those used in [Terzopoulos-84] or [Grimson-81]. While we have not fully explored the computational aspects of these trade-offs, we already know that some of the non-traditional classes are computationally more efficient when employing the semi-reproducing kernel spline algorithm.

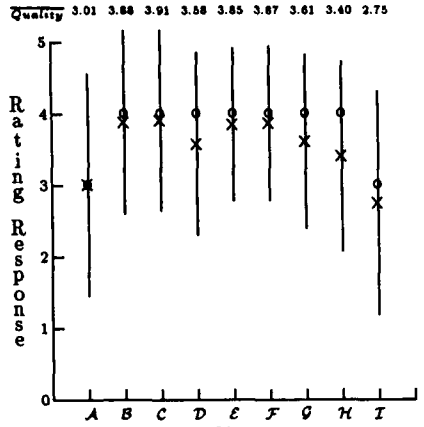


Figure 4: Processed responses for data-sets #1-9. That is, combination over all data-sets.

Secondly, the most appropriate model may be affected by the "smoothness" of the underlying data. If the underlying data contains more than one surface (i.e. contains jump discontinuities), then some of the non-traditional classes were considerably better. If the underlying surfaces were smooth most of the classes tested performed reasonable well.

In the previous sections we pointed out a possible relationship between the number of apparent derivatives in the class and the quality of the associated reconstructions. The exact nature of the relationship seemed dependent on the actual data-set, but it appeared to be a unimodal relationship with the location of the peak depending on the data-set (accounting for differences use to null-spaces).

Finally, for the case when the underlying surface had jump or orientation discontinuities, we showed that changing the norm while keeping the class of functions basically fixed did not greatly affect the quality of reconstructions, while fixing the norm and varying the class of functions did produce significant changes in reported quality.

4 How Contanimated Data Effects the Process

A common technique in solving ill-posed problems (such as the surface reconstruction problem discussed above) is with the use of regularization techniques (see [Poggio-Torre-Koch-85] and the references therein) or smoothing splines (see [Whaba-84]). These approaches however assume that one knows the correct model for the process and provides approaches for dealing with contaminated error.

When modeling a process, it is often as important to model the error as it is to model the underlying process. In doing this, the modeler has two distinct choices – develop a single model that handles both the underlying process and the error, or develop separate models. If one chooses the former, the general approach described above can be applied; however, the derivation of optimal error algorithms becomes more difficult. If one separates the models, one can first experimentally determine the model of the underlying process, then using this may develop algorithms that handle contaminated data.[7]

5 Error, Models and Complexity

Traditionally, one develops a model for a problem and an algorithm which solves it with as small a cost as possible. Such an approach often does not consider what the error inherent in the model is, and how that is related to the error of the developed algorithm. In Section 2, we discussed the general approach for refining the model of a problem using optimal error algorithms. While this may be good for abstract modeling, we realize that an often important aspect of a model is that it can be used to quickly produce some predictions. Thus, one desires algorithms with as small a complexity as possible.

While it is often the case that optimal error algorithms are also optimal complexity algorithms the relationship is not assured, see [Traub-Wasilkowski-Woźniakowski-83]. In particular the complexity depends heavily on the model of computation (especially if parallel computation is to be used), while the error properties are inherent in the problem, independent of any computer. Thus, while the optimal error algorithm generally does not significantly change between the (reasonable) models of computation, the complexity of that algorithm generally does.

However, all is not lost. Once we have developed an appropriate model for a problem, nothing is to prevent us from using that knowledge to develop approximate algorithms which are computationally more efficient than the optimal error algorithm.

References

[Boult-86] Terrance E. Boult. *Information Based Complexity in Non-Linear Equations and Computer Vision.* PhD thesis, Department of Computer Science, Columbia University, 1986.

[Boult-Kender-86] Terrance E. Boult and John R. Kender. Visual surface reconstruction using sparse depth data. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 68–76, June 1986.

[Duchon-76] J. Duchon. Interpolation de fonctions de deux variables suivant le principe de la flexion des plaques minces. *Revue Francaise d'Automatique, Informatique et Recherche Operationnelle*, 5–12, December 1976.

[Grimson-79] W. E. L. Grimson. *From Images to Surfaces: A Computational Study of the Human Visual System.* PhD thesis, MIT, 1979.

[Grimson-81] W. E. L. Grimson. *From Images to Surfaces: A Computational Study of the Human Visual System.* MIT Press, Cambridge, MA, 1981.

[Julesz-71] B. Julesz. *Foundations of Cyclopean Perception.* University of Chiage Press, Chiago, IL, 1971.

[Lee-85] D. Lee. *Contributions to Information-based Complexity, Image Understanding, and Logic Circuit Design.* PhD thesis, Department of Computer Science, Columbia University, 1985.

[7] In the visual reconstruction problems discussed above, we are separating the models and currently researching algorithms for the contaminated case. One obvious way to approach the approximation problem is to use the smoothing spline form of the reproducing kernel spline algorithms.

[Micchelli-Rivlin-77] C. A. Micchelli and T. J. Rivlin. A survey of optimal recovery. In C. A. Micchelli and T. J. Rivlin, editors, *Optimal Estimation in Approximation Theory*, pages 1–54, Plenum Press, New York, 1977.

[Poggio-Torre-Koch-85] T. Poggio, V. Torre, and C. Koch. Computational vision and regularization theory. *Nature*, 317(6035):314–319, 1985.

[Terzopoulos-84] D. Terzopoulos. *Multiresolution Computation of Visible-Surface Representations.* PhD thesis, MIT, 1984.

[Traub-Wasilkowski-Woźniakowski-83] J. F. Traub, G. W. Wasilkowski, and H. Woźniakowski. *Information, Uncertainty, Complexity.* Addison-Wesley,Reading,MA, 1983.

[Traub-Woźniakowski-80] J. F. Traub and H. Woźniakowski. *A General Theory of Optimal Algorithms.* Academic Press, 1980.

[Whaba-84] G. Whaba. Surface fitting with scattered noisy data on euclidean d-space and on the sphere. *Rocky Mountain Journal of Mathematics*, 14(1):281–299, 1984.

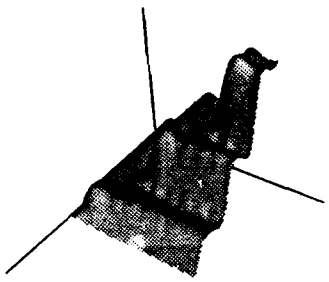


Figure 5: Reconstruction using model $\mathcal{F}$ for data-set # 2.

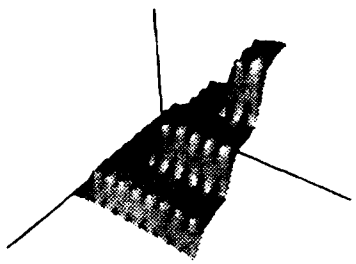


Figure 6: Reconstruction using model $\mathcal{D}$ for data-set # 2.

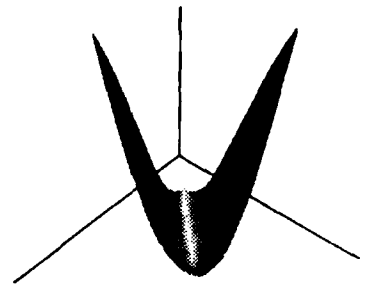


Figure 7: Reconstruction using model $\mathcal{F}$ for data-set # 4.

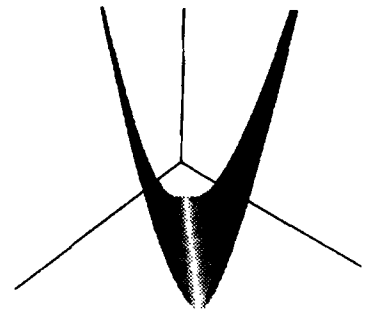


Figure 8: Reconstruction using model $\mathcal{D}$ for data-set # 4.

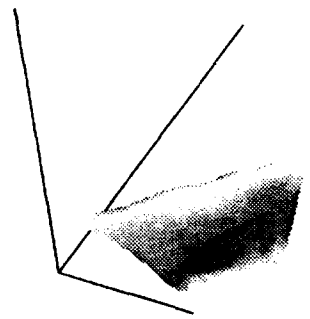


Figure 9: Reconstruction using model $\mathcal{F}$ for data-set # 5

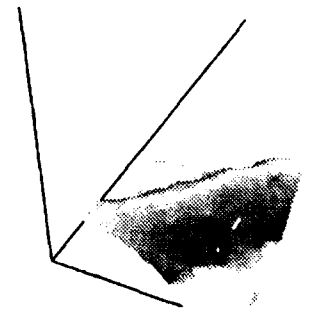


Figure 10: Reconstruction using model $\mathcal{D}$ for data-set # 5

A PARALLEL IMPLEMENTATION AND EXPLORATION OF WITKIN'S SHAPE FROM TEXTURE METHOD

Lisa Gottesfeld Brown and Hussein A. H. Ibrahim*

Department of Computer Science
Columbia University
New York, NY 10027

ABSTRACT

This paper explores A.P.Witkin's shape from texture method [1]. We show the results and performance of this method for a parallel implentation developed on the simulator for the massively parallel tree-structured supercomputer, the NONVON [2]. We show that the parallelization of Witkin's algorithm not only substantially improves the overall efficiency but it does so by exploiting a more powerful representation which could be used by other shape-from methods. Lastly, we show how the accuracy of the method can be improved by using the Gaussian sphere in representing the isotropy assumption and we compare our results with the original method and the more efficient methods developed by Davis, Janos and Dunne [3].

INTRODUCTION

In this paper, we discuss an implementation of A.P.Witkin's algorithm for the recovering of surface orientation from texture for the highly parallel, SIMD, tree-structured NONVON supercomputer. Witkin's approach to the surface from texture problem is well-suited to a wide spectrum of textures and relies upon the assumptions that:

1. The image is an orthographic projection representing a planar surface.
2. The texture itself is considered to be the distribution of edge directions and this distribution is assumed to be uniform prior to the effects of projection. More importantly, the method will work to the extent in which the distribution of edge directions that compose the texture do not resemble the effects of projection.
3. All surface orientations are equally likely. If we consider each surface as being represented by a point on the Gaussian sphere then a given surface is equally likely to be represented by any point on the sphere [4].

Given that these assumptions hold completely, this method has already been proven to be quite robust. The major drawbacks are that it fails when confronted with textures with regular features (assumption 2) and it is computationally inefficient. L.S. Davis, L. Janos and S.M. Dunn have proposed two algorithms which numerically approximate Witkin's method and speed the computation by as much as a factor of four. In this paper, we consider ways to improve the efficiency through parallelization. We take care to use representations which give accurate measurements and can be used to extend the method to a larger domain.

The idea behind Witkin's method is as follows. We assume that the distribution of edge directions is uniform prior to the projection. If the surface is slanted about an axis parallel to the X-axis by an angle θ with respect to the image plane then we expect the distribution of edge directions to become dominated by edges whose directions are close to horizontal. The larger θ is, the greater the domination of these edges. This makes intuitive sense since all the edges in the surface are being more and more compressed along their vertical component. If the surface is slanted about a different axis which makes a tilt angle τ with the X-axis, then the distribution is just shifted. Examples of these distributions for various slants and tilts are shown in Figure 1.

L.Wolff [5] presents an outline for the implementation of this method on the NONVON which forms the basis for the algorithm used here. The three phases of the algorithm are:

1. The determination of the edges in the image in each of the edge directions considered.
2. The histogramming of the edge directions.
3. Computing the likelihood for the histogram for each possible surface orientation and finding the maximum likelihood.

The first section of this paper discusses the first two phases of the algorithm while the second section elaborates upon the the third phase. The final section shows how the accuracy of the method can be improved by using a better representation.

THE EDGE DIRECTIONS

This is computationally the most important phase of the algorithm. Since this phase is only the input into the algorithm it is often overlooked; but the choice of representation and method used here greatly effect the types of information available for subsequent use. Once the histogram of edge directions is found the computation of the likelihood estimates is well defined and computationally less intensive. However, there are several different ways to determine the edges and their orientations and we would

*This work supported in part by DARPA grant # N00039-84-C-0165 and DARPA grant # DACA76-86-C-0024

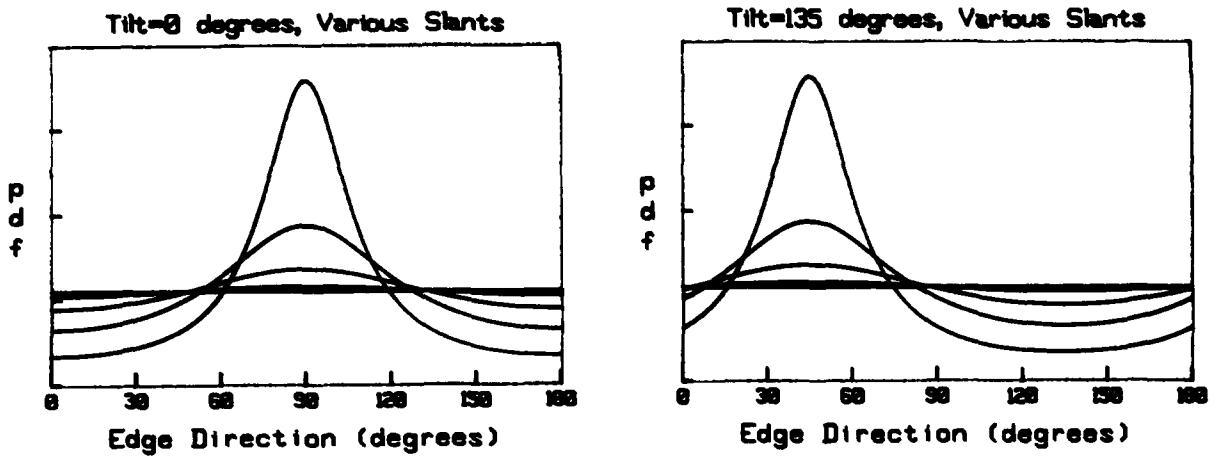


Figure 1. Plots of $pdf(edge\ direction|\theta,\tau)$ for $\tau = 0$ (left) and $\tau = 135$ (right) at several values of θ.

like to do this both efficiently utilizing our parallel architecture and with an eye to facilitating the detection of edges which do not satisfy the assumptions.

For the initial detection of the edges, Witkin proposed the Marr-Hildreth zero-crossing operator which he used successfully to demenstrate his algorithm. This method is also applied here using the parallel functions for the NONVOVN written by Burton, Lee, and Wolff[6]. Their implementation is designed about a multiresolution image pyramid for the purpose of stereopsis. With this arrangement of the image data, their implementation could exploit the processing tree architecture of the NONVON. The NONVON supports four modes of communication which are: global,linear,binary tree and within the lowest level of the tree, there is mesh communication. Hence, the multiresolution image pyramid is stored in every other level of the tree and intra-level communication is optimal for the lowest level image. For two pixel neighbors, PE1 and PE2 at a higher level however, data must first be passed from PE1 down the tree to the mesh level, cross over to the appropriate child of PE2 and then ascend the tree before reaching PE2. Having a multiresolution pyramid for the stereopsis was useful for performing multi-level matching but it could similarly be used in improved edge detection. Marr and Hildreth [7] develop this idea. Each level is used to find the zero-crossings for differing frequency bands and then a set of parsing rules determine the relationship between the zero-crossings at each level. The parsing rules are determined by physical constraints such as the spatial localization of intensity changes due to each physical phenomena in the visual world. Thus, zero-crossings are expected to be found at each level unless more than one phenomena have been combined from a higher frequency band. From these rules, the primitives of the primal sketch are found. This same technique might by exploited for use with Witkin's algorithm. Distribution of edge directions found at each level could be analysed together with the parsing rules to determine the frequency band which give the most reliable results or greatest likelihoods. In fact, one of the problems encountered with zero-crossings is that more than what is normally percieved as the edges in the image are detected. This is significant since the set of edges which are optimal for Witkin's method are not known. In our current implentation however, just the mesh level is used for edge detection so that each convolution requires only $O(N^2)$ operations where the convolution mask is NxN and the computations to find zero-crossings in each direction requires $O(S)$ in the worst case (where the image is SxS) and $O(k)$ in the average case, for some small constant k.

Given this technique for edge detection, it still remains to determine the directions of the edges. To do this, let us first consider the principles behind our edge detection scheme. The Marr-Hildreth operator was selected as the optimal smoothing filter that satisfied two physical constraints. The first constraint is one of spatial localization since it is believed that only nearby points contribute information to the intensity changes at any given point. The second constraint is a frequency localization. We would like a smooth and band-limited filter. These constraints lead directly to the D^2G operator or to finding the zero-crossings of the second directional derivative of the image convolved with a Gaussian filter. This is approximated by taking the Laplacian of the Gaussian filtered image or $\Delta^2 G$.

The first thing to note is that finding a zero-crossing in a given direction does not necessarily tell us about the direction of an edge at this point. Indeed, it is possible that there is a zero-crossing in *every* direction at a given point when for example, there is only a single edge formed by a uniform intensity change with constant lines that run parallel to the edge. To choose among several zero-crossings at one point, we have two simple strategies:

1. Choose the zero-crossing which has the maximum slope.
2. Choose the zero-crossing whose orientation agrees with the orientations of other neighboring zero-crossings.

The second strategy is supported by the theorem that Marr and Hildreth call, the *condition of linear variation* in which for smoothed images (as in our case) the two strategies above are equivalent. The theorem tells us that the intensity variation near and parallel to the line of zero-crossings should be locally linear. We are ultimately interested in the direction of an edge at this point, and we see that according to this theorem the line of zero-

crossings is in the direction of the edge since we expect linear intensity changes along and parallel to the edge and a maximum change across it.

Thus, there are two approaches that can be taken in extracting the edge directions from the zero-crossings, both which can be easily implemented in our parallel scheme. The first approach takes the edge direction as orthogonal to the orientation of the zero-crossing with the maximum gradient. The second approach considers an edge only if there are two zero-crossings of the same orientation which are adjacent to each other on the line perpendicular to the zero-crossing orientation. The direction of this edge is then the direction formed by the line on which the two zero-crossings lie. This method has the advantage that an edge is more clearly part of contour and less likely to have been contributed by noise. On the other hand, since we are only interested in "texture," a contour may be irrelevant. Finally, an edge in the vertical or horizontal direction is more easily found due to the nature of the grid and this will skew our histogram values.

For both approaches we are limited by four directions in which we usually look for zero-crossings given an image grid where each pixel has neighbors along four lines: vertical, the two diagonals and horizontal. For histogramming edge directions, four directions can suffice depending on the accuracy desired in the results. We have found that using four directions with accurate information is all that is needed to determine which of 36 uniformly spaced (on the Gaussian sphere) surface orientations for an image that is only 32x32. But for more precise results, we need to consider ways of refining our methods. For the first approach, we can consider the gradients of perpendicular zero-crossings as orthogonal components of a vector which determines the orientation of the maximum slope and thus is orthogonal to the edge direction. There are actually two pairs of orthogonal components that we can use as redundant information and determine the direction of the edge with the accuracy affordable by the data. For each pair, if only one zero-crossing is found the gradient for the other is considered to be zero. If both pairs have at least one zero-crossing then the two resulting vectors can be averaged. If the two resulting vectors are more than 90 degrees apart then the information at that pixel is thrown out since it is contradictory.

For the second method, we can consider triplets or quadruplets of zero-crossings in order to increase the number of different directions that we can find for the edges but this requires even longer edges. Furthermore, the distribution of detectable edge directions continues to be nonuniform. For greater discernment in the edge directions the first method is superior for our purposes since we can uniformly histogram the edge directions which are found in a single non-expanding set of operations. However, if the vision system incorporates a contour following algorithm, then this method might serve us well. Small contours could be used for the edge direction distribution and larger contours, indicative of non-textural phenomena, could be passed on to other modules.

Once the edge directions have been found, we can take advantage of the NONVON tree communication capability to compute the histogram. Since the edge directions now reside at the mesh level, we simply accumulate the number of edges in each direction as we ascend the tree. The algorithm will require only $O(logn)$ operations where n is the number of pixels in the image as opposed to $O(n)$ needed by a sequential machine.

There are of course several alternatives to this scheme for determining the edge directions. Two methods that we have considered are: (1) using the gradient determined by the orthogonal Sobel operators and (2) using the wedge shaped regions centered at the origin of the power spectrum of the image. The former can be implemented on the NONVON in a similar fashion as the Laplacian of the Gaussian since it is a convolution and it has the advantage of measuring edge orientation more accurately. The latter also has a well-studied parallelization and although it is not as efficient as the convolution with a small mask, the histogram could be found directly. For all of these methods, there is an interesting threshold parameter which can be used like the bandwidth to determine the optimum maximum likelihood.

ORIENTATION LIKELIHOODS

This phase of the algorithm although computationally somewhat complex, is not nearly as computationally intensive and therefore we initially thought that it did not justify implementation on the NONVON but could more easily be accomplished by its host computer. Basically, it consists of computing the likelihood of each possible orientation given the distribution of edge directions. While the number of possible orientations is infinite, they can be well represented by a small sized subset on the order of a hundred. However, after considering all the benefits, it becomes clear that it is worthwhile to take advantage of the parallel NONVON architecture.

The likelihoods that we need to compute are:

$$L(\theta,\tau|A^*) = \frac{sin\theta}{\pi}\prod_{i=1}^{n}\frac{\pi^{-1}cos\theta}{cos^2(\alpha_i^*-\tau)+sin^2(\alpha_i^*-\tau)cos^2\theta}$$

where θ,τ are the slant and tilt and $A^* = [\alpha_1^*,...,\alpha_n^*]$ are the histogram values. This equation is an application of Bayes Theorem where the first term is the $pdf(\theta,\tau)$ and the rest is the $pdf(A^*|\theta,\tau)$. Notice that the likelihood of a particular surface orientation depends on the probability that *that* surface occurs. We assume that all surfaces are equally likely, but this does not mean that each pair of slant and tilt values are equally likely. The probabilty that a surface is not slanted at all is very small compared to the probability that it is slanted 90 degrees because of the range of tilts that can occur in the latter case. This plays an important role in determining the surface orientation of maximum likelihood (as we shall see later) and whenever possible it should be modified to reflect more pertinent knowledge of the environment or information obtained by other shape-from methods.

Davis, Janos and Dunne have shown two ways to find the maximum likelihood more efficiently by (1) using a good geometric heuristic to estimate the tilt and then using a one-dimensional Newton method to obtain the slant and (2) expressly computing the orientation by accurately approximating the equations. Both methods do indeed improve the efficiency (the second one, very effectively) although with some loss of accuracy. On the other hand, much relevant information is lost especially if we would like to use this method in conjunction with other shape-from techniques. Specifically, we no longer know the likelihoods at each orientation. Witkin showed in his original paper the iso-density contour maps of the orientation density functions obtained for images of geographic contours. These figures show graphically

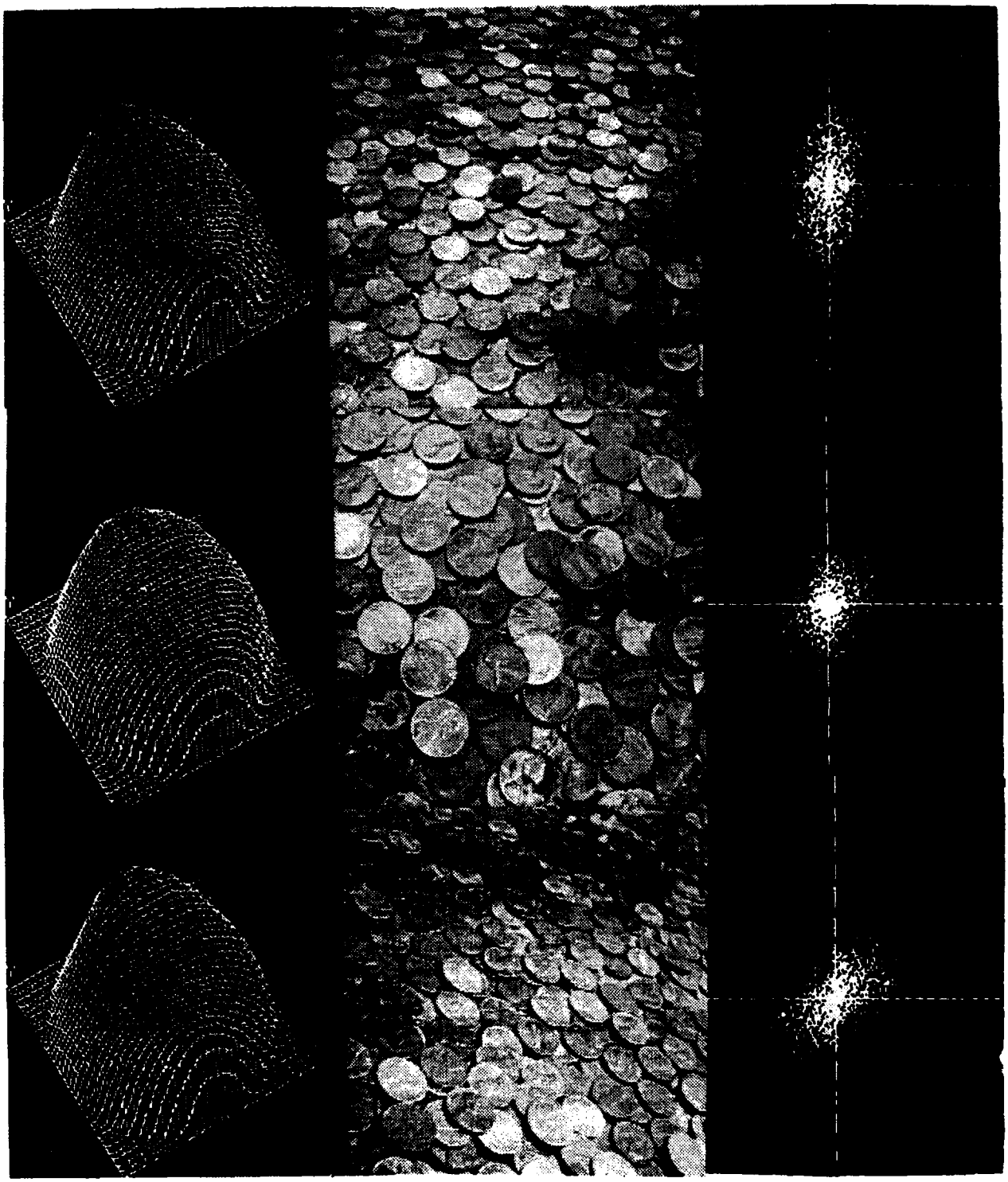


Figure 2. The center column contains the images whose orientations are, top: $\tau = 90, \theta = 45$, center: $\tau = 90, \theta = 20$, bottom: $\tau = 45, \theta = 45$. The left column contains the likelihood at each surface orientation where where the distance from the origin is θ and the angle is τ. The right column contains the Fourier spectra.

much more information that can be gathered from a single maximally likely orientation. Not only are other likely orientations exposed but the sharpness of the peaks, unlikely orientations, and the overall shape of the confidence measures are seen. For this reason, and because of the non-trivial speed-up, we decided to parallelize this stage of the algorithm, one PE for each likelihood, so that a fine grained output of the orientation likelihoods could be obtained. Once these likelihoods are found, the maximum can be found by traversing up the tree. But in addition, we have created an ideal structure for outputting *all* the likelihoods to other shape-from methods and for feedback between the likelihoods and the edge direction distributions found at varying bandwidths and thresholds. This latter result, can be used to identify the edges which optimally satisfy the assumptions, allowing Witkin's method to be used on more natural textures successfully. A sample of our results are given in Figure 2.

IMPROVING THE ACCURACY

The assumption that all surfaces are equally likely is called the isotropy assumption and from it the joint pdf of the slant and tilt is derived. Using the pdf that Witkin derived, given above, gives poor results because of the singularly small probability associated with a surface lying close to the image plane.[1] This causes an error in the predicition of the surface orientation whenever the slant is small. This error arises because of the choice of the coordinate space used to represent orientation. Furthermore, if we take equally spaced intervals of θ and τ the probability associated with each grid square is not uniform and thus the discrimination of surface orientation is also not uniform.

All of this can be easily remedied by a more appropriate choice of the coordinate space representation of orientation. The singularity that occurrs when the slant is small is a result of the nonuniform probability associated with each orienation region, the regions being smallest when the slant approaches zero. If instead of selecting equally spaced intervals of θ and τ, we choose equal size areas on the Gaussian Sphere, the joint pdf is simply a uniform distribution and thus a constant. The advantages are twofold: large errors at small slants are avoided and the resolution of the results becomes more consistent.

Figure 3 shows the decrease in the error as a function of slant by transforming the orientation space into Gaussian Sphere. Ten random samples of 2000 uniformly distributed tangent directions were projected at a constant tilt for each slant. The projected tangent directions are then histogrammed and given as input into Witkin's original method which finds the maximum likelihood of 100 surfaces: 10 slants and 10 tilts, equally spaced and into the modified method which finds the maximum likelihood of 100 surfaces approximately uniformly spaced on the Gaussian Sphere. The error in the predicted orientation (θ^*, τ^*) from the known orientation (θ, τ) is measured in two ways. The bottom plot shows the rms error similar to that used by Davis, Janos and Dunne defined by,

$$\epsilon(rms) = \sqrt{(\theta - \theta^*)^2 + (\tau - \tau^*)^2}$$

where the difference between two tilt angles is the minimum of the possible differences of the two angles modulo π. Notice the inordinately large errors at small slants for both methods. This is in part because at small slants the effects of the direction of slant, namely the tilt, are neglible. (See Figure 1.) On the other hand, a large tilt error at small slants does not indicate that a large rotation is needeed to orient the actual surface to the predicted one and yet the size of this rotation is more representative of the error that we percieve. Therefore, an improved measure of the error would be the angular distance separating the actual surface $\bar{O}$ from the predicted surface $\bar{O}^*$ where both are represented by three dimensional cartesian vectors on the Gaussian Sphere:

$$\epsilon(sphere) = arccos(\bar{O} \cdot \bar{O}^*).$$

The top plot in Figure 3 shows the same errors measured in this way. There is still a large error associated with small slants although this better represents the error we would percieve. It is not surprising these errors are still large since changing the orientation of a surface at small slants changes the region that is percieved only slightly while at large slants, the size and location of the physical region seen changes drastically.

[1] Lee and Rosenfeld [8] have corrected this to account for the discrete nature of the data and concluded that the pdf should be $(1/2\pi)sin2\theta$ but this does not alter the problem mentioned here.

The implementation of this method on the simulator for the NONVON computer, has exposed both the advantages and limitations of Witkins's statistical approach to texture. One obvious limitation is that not all surface orientations can be distinguished. For every surface orientation that the method finds, there is another orientation which slants in the opposite direction which is equally likely. This is obviously counter to human perception. It suggests that this method is somehow incomplete or the wrong approach. However there are several reasons why this method is appealing. Most other methods which derive shape from texture rely upon gradient measures. These methods rely upon determining texel size, shape or spacing and assumptions concerning their uniformity. These methods are complicated by determining or locating texels and limited to very specific domains. Notice how these methods contrast with Witkin's approach which will often fail when their assumptions hold but work when they fail. For example, uniformly shaped texels will cause Witkin's method to fail if the texel shape is biased towards certain directions but when the texels are all shaped differently Witkin's is likely to be applicable. Thus, it would be useful, if Witkin's method could be applied so that instead of solely determining the orientation which is most likely to be represented by the given edge distribution, the likelihood of each orientation was used as information to be passed onto to other shape from texture methods. These other methods which measure shape from texture using gradients, can then determine precisely what Witkin doesn't: the local directionality which uniquely determines the surface orientation.

References

[1] A. P. Witkin, "Recovering Surface Shape and Orientation from Texture," *Artificial Intelligence* **17** (1981), 17-45.

[2] D. E. Shaw, "Organization and Operation of a Massively Parallel Machine," to appear in *Advanced Computer Technology*, Guy Rabbat, editor, Van Nostrand Reinhold, 1986.

[3] L. Davis, L. Janos, and S. Dunn, "Efficient Recovery of Shape from Texture," *IEEE Trans. on Pattern Analysis and Machine Intelligence PAMI-5* No. 5, Sept. 1983, 486-492.

[4] J. R. Kender, "The Gaussian Sphere: A Unifying Representation of Surface Orientation," Department of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213.

[5] L. Wolff, "Parallel Algorithms for the Determination of Shape from Texture," Research Project report under Hussein Ibrahim, Computer Science Department, Columbia University, New York,NY 10027.

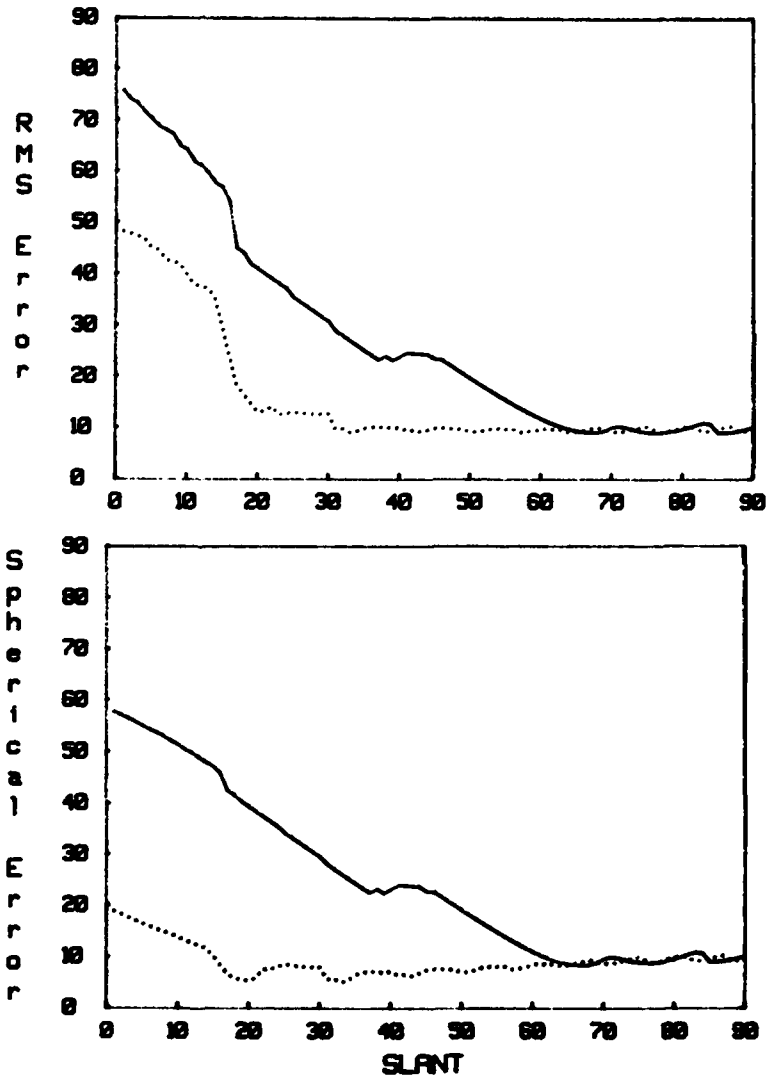
[6] T. Burton,A. Lee, and L. Wolff,"Parallel Stereo Algorithms: A Multiresolution Approach," Research Project report under Hussein Ibrahim, Computer Science Department, Columbia University, New York,NY 10027.

[7] D. Marr and E. Hildreth, "Theory of Edge Detection," *Proc. Royal Soc. London* **B207**, 1980, 187-217.

[8] C. H. Lee and A. Rosenfeld, "Improved Methods of Estimating Shape from Shading Using the Light Source Coordinate System," University of Maryland Computer Science Technical Report TR-1277, April 1983.

Figure 3.

Comparison of Witkin's Original Method and Modified Version (dotted line)



LOCALIZED INTERSECTIONS COMPUTATION FOR SOLID MODELLING WITH STRAIGHT HOMOGENEOUS GENERALIZED CYLINDERS

Jean Ponce and David Chelberg

Stanford Artificial Intelligence Laboratory
Stanford University, Stanford, Ca 94305

Abstract: This paper reports progress in the development of a solid modelling system combining straight homogeneous generalized cylinders through set operations. Two basic components of this system are the modules which compute the set operations between primitives and display the resulting solids using ray tracing. These two modules are also very computationally intensive as they involve a large number of surface-surface and ray-surface intersections computations. We introduce a novel hierarchical representation for straight homogeneous cylinders called *Box Tree*. The Box Tree is analogous to a Quadtree in parameter space. It is an exact boundary representation which describes the surface of the associated generalized cylinder by a hierarchy of enclosing boxes. We use the Box Tree to efficiently compute the set operation and ray tracing algorithms by localizing the search for intersections to the regions where they may occur. We discuss complexity issues and illustrate the performances of our modelling system on a variety of examples.

Introduction

A *generalized cylinder* is the solid obtained by sweeping a surface, its *cross section*, along a curve, its *axis*, or *spine*. The axis is not necessarily straight, or even planar; the cross section is not necessarily circular, or even constant; its deformation along the axis is governed by a *sweeping rule*. Generalized cylinders were intended to represent primitive solids, while complete shapes would be represented as part-whole graphs of joined primitives [2]. Generalized cylinders have been extensively used to represent 3D objects in computer vision [13],[15],[23]. The most successful vision system to date using generalized cylinders as its primary representation for three dimensional objects is probably *Acronym* [3].

Even in Acronym, however, only very restricted subclasses of generalized cylinders are implemented (circular or simple polygonal cross section, straight or circular spine, linear or bilinear sweeping rule). Very restricted joining operations are considered. In fact, subparts are simply affixed to each other. On the other hand, solid modelling systems have used for years complex joining operations, such as set operations [19],[25], or blending of surfaces [7]. The primitive solids are in general simple (polyhedra bounded by a few planar or quadric faces), but set operations now extend to more complex primitives, such as polyhedra with many planar faces [12],[18], or solids bounded by parametric surface patches [4].

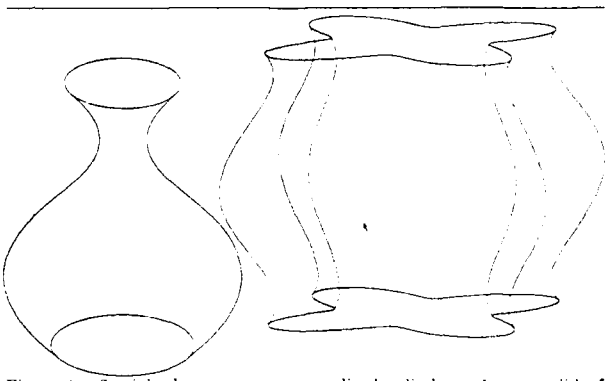


Figure 1. Straight homogeneous generalized cylinders. 1-a: a solid of revolution, 1-b: a SHGC with a star shaped cross section and a sweeping rule which is a polynomial of degree 4.

This paper reports progress in the development of a geometric modelling system which manipulates a wide class of generalized cylinders with the power and flexibility of a CAD system. Our primitives are straight homogeneous generalized cylinders (SHGC's, see [23]). They are generalized cylinders obtained by scaling a reference cross section along a straight axis (Figure 1). In the modelling system presented here, the scaling function is an arbitrary continuous (C^0) and piecewise continuously differentiable (C^1) real function. The cross section is an arbitrary star shaped C^0 and piecewise C^1 curve (see Section 2, Figure 4).

The primitives are joined through set operations (union, intersection, difference). *Ray tracing* [20],[26] is used for hidden line/surface display of the resulting composite objects. Set operations and ray tracing algorithms are tradi-

Support for this work was provided in part by the Air Force Office of Scientific Research under contract F33615-85-C-5106 and by the Advanced Research Projects Agency of the Department of Defense under Knowledge Based Vision contract AIADS S1093-S-1.

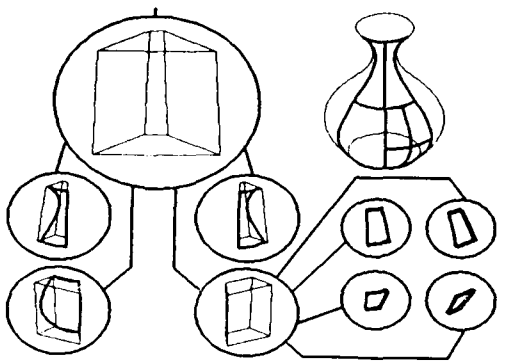


Figure 2. The Box Tree representation. to each quadrant of a Quadtree in parameter space we associate a box which encloses the associated sector. At the leaf level, the nodes contain also a representation of the sectors themselves

tionally very computationally expensive as they involve a large number of complex surface-surface and ray-surface intersections computations. This is more generally true of any intersection algorithm which manipulates complex surfaces. A recent approach [4],[10],[12],[18] for speeding up these algorithms has been to use a hierarchical representation of surfaces to efficiently *localize* the search for geometric intersections to the regions where they may occur.

In this paper, we follow an analogous approach. We introduce a novel hierarchical representation for SHGC's called the *Box Tree* (Figure 2). It corresponds to a Quadtree [22] in parameter space and represents the associated SHGC by a hierarchy of boxes enclosing surface patches called sectors. The Box Tree is an exact representation (i.e. it does not involve any approximation of the object). We use it to localize efficiently the set operation and ray tracing algorithms. We discuss the localized intersections scheme in more detail in Section 1. We define in Section 2 a SHGC's sector, the associated enclosing box, and build the Box Tree. In section 3 and 4 we give the localized set operation and ray tracing algorithms. Complexity issues are discussed. Both algorihms have been implemented and are illustrated in Figures 3,6,8,9 and 12.

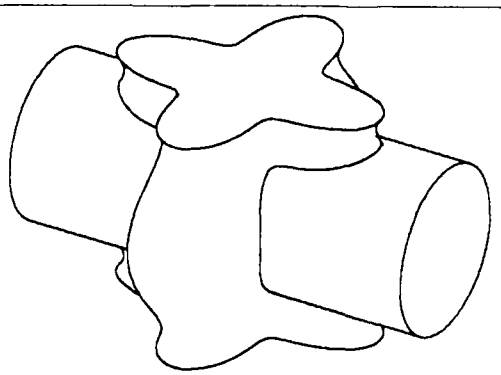


Figure 3. The display of the union of a cylinder and a polynomial SHGC using ray tracing at the limbs.

1. Localized intersection algorithms

Set operation algorithms for boundary representations of solids can be decomposed into the following steps.

[1] Compute the list of all the intersecting faces of both solids, and associate to each of them the list of faces it intersects.

[2] Use this list to compute the intersection curves and cut the intersecting faces into non intersecting subfaces along these curves.

[3] The intersection curves partition the surfaces of both solids into a set of connected components (CC's) of non intersecting faces. Classify the CC's of each solid as being inside or outside of the other solid.

The resulting solid is formed by the appropriate combination of inside and outside faces of both solids (e.g. the union is made of all the outside faces). Let S_1, S_2, and S_3 be the complexities of the three steps. The overall complexity is $S = max(S_1, S_2, S_3)$.

Similarly, ray tracing can be decomposed for each pixel into the following steps:

[1] Compute the list of all the faces intersecting the ray associated to the pixel.

[2] For each face in this list, compute the intersection point. Find the first point along the ray and use it to compute the intensity of the pixel.

The overall complexity of ray tracing per pixel is $R = max(R_1, R_2)$ where R_1 and R_2 are the complexities of the two steps.

Suppose now that the solids' surfaces are represented by polyhedra whose n (possibly curved) faces all have comparable sizes. To find the intersecting faces of two solids, we must compare all faces of both solids, so S_1 is $O(n^2)$. S_2 is proportional to the length of the list of intersecting faces. For homogeneous decompositions, this length is $O(p\sqrt{n})$, where p is the perimeter of the intersecting curves (the area of a surface grows as the square of the length of the curves drawn on it). Finally, for each connected component CC_i, step 3 involves classifying one face as inside or outside ($O(n)$ step, see for example [11]), and then spreading this classification to all the faces of CC_i ($O(n_i)$ step, where n_i is the number of faces of CC_i, if we suppose that accessing the neighbors of a face can be done in constant time). It follows that S_3 is $O(c.n)$ where c is the number of CC's. p and c are essentially constants, so the overall complexity of the set operation algorithm is $O(n^2)$. Similarly, for ray tracing all faces must be tested against a ray, so R_1 is $O(n)$. R_2 is proportional to the essentially constant number r of faces actually intersecting the ray, so R is $O(n)$.

In the context of set operations, several authors (see for example [4],[12],[18]) have suggested to use hierarchical surface representations to *localize* the search for intersect-

ing faces to the regions where they may occur, i.e. near the intersection curves. Using this method, they have been able to reduce the overall complexity of the set operation algorithm to $O(n)$. Mantyla and Tamminen [12] organize the space around solids in a structure called *BOX-Excell*, and use it to efficiently access all faces intersecting a given face. Carlson [4] and Ponce and Faugeras [18] use hierarchies of enclosing boxes (Quadtree and Prism Tree data structures) to prune the search for the intersecting faces. Similarly, it is possible to use the Prism Tree representation for polyhedra, or a similar representation for fractal surfaces [6],[10] to reduce the ray tracing complexity R to $O(\sqrt{n}\log n)$. In the next Section, we introduce the Box Tree representation for SHGC's. We will show in Sections 3 and 4 how we can use it to localize the set operation and ray tracing algorithms and reduce their respective complexities to $O(\sqrt{n})$ and $\log n$, where n is the number of faces in the equivalent homogeneous decomposition.

2. Sectors, Boxes, and the Box Tree

We now define sectors and boxes (Figure 5) and build the Box Tree (Figure 2). A point on the surface of a straight homogeneous generalized cylinder is given by the following equation.

$$\overrightarrow{OP}(\theta, z) = r(z)\rho(\theta)(cos\theta\vec{i} + sin\theta\vec{j}) + z\vec{k};$$

$$(\theta, z) \in [0, 2\pi] \times [z_-, z_+] \quad (1)$$

The function $\rho(\theta)$ specifies the shape of the cross section of the SHGC, while $r(z)$ is the scaling *sweeping rule* of the SHGC. Notice that this formulation assumes that the cross section is star shaped (Figure 4). In the sequel we suppose that $\rho(\theta)$ and $r(z)$ are strictly positive, C^0, and piecewise C^1.

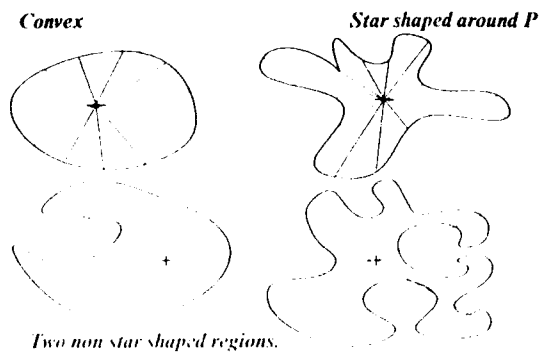


Figure 4. A region is star shaped with respect to a point P iff for each point on its boundary, the whole segment between P and the point is included inside the region. Convex regions form a strict subset of star shaped regions.

Definition: the sector of a straight homogeneous generalized cylinder associated to a quadrant $Q = [\theta_1, \theta_2] \times [z_1, z_2]$ is the set of points $P(\theta, z)$ such that $(\theta, z) \in Q$.

For star shaped cross sections, it is straightforward that a sector is a connected part of the generalized cylinder surface, completely included within the planes $z = z_-$, $z = z_+$, $\theta = \theta_1$ and $\theta = \theta_2$. Let A, B, C, and D denote the four corners of the sector, i.e. the points defined respectively by (θ_1, z_1), (θ_2, z_1), (θ_2, z_2) and (θ_1, z_2). It is straightforward that these four points are in a same plane Π. Using two planes parallel to Π we can build a box enclosing the sector.

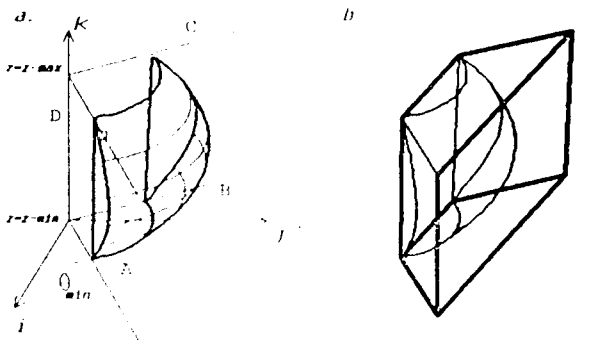


Figure 5. A sector and the associated box. The box is defined by two horizontal planes, two vertical planes, and two planes parallel to the quadrilateral (A, B, C, D).

Definition: the box associated to a sector is the convex polyhedron limited by the planes $\theta = \theta_1$, $\theta = \theta_2$, $z = z_1$, $z = z_2$, and the two planes Π_1 and Π_2 parallel to Π such that their distance from Π corresponds to the negative minimum and positive maximum of the distance of a sector point to Π.

The distance between the planes Π_1 and Π_2 is called *thickness* of the box in the sequel. It characterizes the tightness of the fit of the box to the associated sector. It is straightforward that a sector is entirely enclosed into the associated box. This implies that if two boxes don't intersect, then the associated sectors don't intersect either. Conversely, if two sectors intersect, then the associated boxes intersect too.

We now show how to compute Π_1 and Π_2, i.e. how to compute the extrema of the (signed) distance d between a point P and the plane Π. d is given by

$$d(P, \Pi) = \frac{\overrightarrow{P_1P} \cdot \vec{n}}{\|\vec{n}\|} = \frac{e(P, \Pi)}{\|\vec{n}\|}$$

where P_1 is a point in the plane and $\vec{n}$ is a (non necessarily unit) vector orthogonal to Π. It is extremal at any point where the numerator $e(P, \Pi)$ is extremal. We call e the denormalized distance in the sequel. If we choose for example $P_1 = A$ we get after some algebraic manipulation

$$\vec{n} = (z_2 - z_1)[(\rho_2 sin\theta_2 - \rho_1 sin\theta_1)\vec{i}$$
$$- (\rho_2 cos\theta_2 - \rho_1 cos\theta_1)\vec{j}]$$
$$- (r_2 - r_1)\rho_1\rho_2 sin(\theta_2 - \theta_1)\vec{k}$$

And

$$e(P,\Pi) = r\rho(z_2 - z_1)[(\rho_2 sin(\theta_2 - \theta) - \rho_1 sin(\theta_1 - \theta)] + \rho_1\rho_2 sin(\theta_2 - \theta_1)[r_2(z_1 - z) - r_1(z_2 - z)]$$

We suppose from now on that both ρ and r are C^1 within the sector. The local extrema of d are located at the points where both the partial derivatives of e are 0. Let us write these derivatives.

$$\frac{\partial e}{\partial \theta} = r\rho'(z_2 - z_1)[\rho_2 sin(\theta_2 - \theta) - \rho_1 sin(\theta_1 - \theta)] - r\rho(z_2 - z_1)[\rho_2 cos(\theta_2 - \theta) - \rho_1 cos(\theta_1 - \theta)]$$

$$\frac{\partial e}{\partial z} = r'\rho(z_2 - z_1)[\rho_2 sin(\theta_2 - \theta) - \rho_1 sin(\theta_1 - \theta)] - (r_2 - r_1)\rho_1\rho_2 sin(\theta_2 - \theta_1)$$

As r is strictly positive, the local extrema of d are finally given by the points (θ, z) which verify the two equations:

$$\rho'[\rho_2 sin(\theta_2 - \theta) - \rho_1 sin(\theta_1 - \theta)] = = \rho[\rho_2 cos(\theta_2 - \theta) - \rho_1 cos(\theta_1 - \theta)] \quad (2)$$

$$r'\rho(z_2 - z_1)[\rho_2 sin(\theta_2 - \theta) - \rho_1 sin(\theta_1 - \theta)] = = (r_2 - r_1)\rho_1\rho_2 sin(\theta_2 - \theta_1) \quad (3)$$

Notice that Equation (2) is an equation in θ only. It can be solved numerically on the interval $[\theta_1, \theta_2]$. Once the solutions have been computed, the corresponding values of θ can be substitued in (3), which becomes a one parameter equation in z. It is solved the same way. If no extremum is found inside the sector, we must look for extrema in one of the planes $\theta = \theta_1$, $\theta = \theta_2$, $z = z_1$, and $z = z_2$. This is done by substituting the right value of z or θ in (2) or (3), and solving this equation. Once the points for which the distance is extremum are found, the actual distance is computed from the denormalized distance by dividing it by the norm of $\vec{n}$.

We can now define the Box Tree as a hierarchy of boxes.

Definition: the *Box Tree* (Figure 2) associated to a quadrant Q_0 of a straight homogeneous generalized cylinder is a tree of degree 4 where each node N is associated to a subquadrant Q of Q_0 in parameter space and contains a pointer to the associated box B. The root is associated to Q_0. If a node N is not a leaf, it has four sons that correspond to the four subquadrants of Q. The leaves of the tree contain a pointer to the associated sectors.

The easiest way to build the Box Tree associated to the surface of a SHGC as defined in (1) would be to associate the root of a Box Tree to the quadrant $Q_0 = [0, 2\pi] \times [z_-, z_+]$ and then subdivide it normally. Unfortunately, this is impossible, as the box associated to a sector is defined only if $\theta_2 - \theta_1$ is not π or 2π (otherwise the planes $\theta = \theta_1$, $\theta = \theta_2$ and Π are parallel). So we choose to associate only three sons to the root of the Box Tree BT_0 corresponding to Q_0. They are associated to the three quadrants $[0, 2\pi/3] \times [z_-, z_+]$, $[2\pi/3, 4\pi/3] \times [z_-, z_+]$, and $[4\pi/3, 2\pi] \times [z_-, z_+]$. In the case where ρ or r are only piecewise C^1, the root points to as many sons as there are quadrants where ρ and r are both C^1.

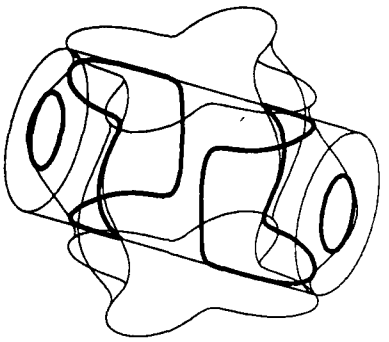


Figure 6. A polynomial SHGC, a cylinder, and their intersection curves.

Each of the ends of a SHGC is represented by the one-dimensional equivalent of the Box Tree, a tree of degree 2 obtained by recursive subdivisions of θ intervals. Let BT_1 and BT_2 be these two trees, the root of the Box Tree BT associated to a SHGC points to BT_0, BT_1, and BT_2. A simple enclosing box (sphere and discs) is associated to BT and each BT_i. In the complexity analyses of Section 3 and 4, we will suppose that Box Tree nodes are always subdivided homogeneously into four subnodes. We have just seen that this is only an approximation at the root level of the tree, but it doesn't change the overall complexity of the algorithms.

Carlson [4] and Ponce and Faugeras [18] have used very similar representations for solids bounded by parametric patches [5] and polyhedra. In these representations however, the total number of nodes at the leaf level is fixed in advance. It is simply the total number of faces (parametric patches or planar polygons). In our case, the SHGC is entirely specified by one analytic expression, and we don't have to subdivide its surface uniformly into sectors. Instead, the surface is subdivided adaptively, and the Box Tree is built at the same time, during the execution of the intersection algorithms. This is especially important for set operations (see Section 3).

The Box Tree is an exact representation: it follows from the fact that the sectors associated to the leaves of the tree partition exactly the surface of the SHGC. No approximation is made during the localization step of the Box Tree algorithms. Approximations only occur during the (relatively cheap) second steps of both set operation and ray tracing algorithms (see Section 3 and 4). In the current implementation, the sectors are then replaced by approximating planar faces (the quadrilaterals $ABCD$). Some numerical scheme could be used instead, so this step could be tuned to achieve a given accuracy without recomputing the localization step. This is an advantage of the Box Tree over homogeneous decompositions, where the resolution is fixed in advance.

A last remark about the Box Tree: it is the world space counterpart of a Quadtree [8],[21],[22],[24] in parameter space, "enriched" by some geometric information. This allows us to use the classical analysis of Quadtree algorithms to get an estimate of the Box Tree algorithms complexity. This will prove especially useful for set operations (see next Section). Notice however that although the Box Tree is similar to the Quadtree in parameter space, it is very different from the Octree related data structures [9],[14] which generalize the Quadtree to 3D space. The Octree directly represents the volume of an object and is attached to a given reference frame, while the Box Tree is a boundary representation, and is attached to the object itself (in world space).

3. The set operation algorithm

We follow the basic algorithm described in Section 1. The only difference with homogeneous decompositions is in the first step, finding the intersecting faces. The algorithm is a direct generalization of the Strip Tree [1] and Prism Tree [18] algorithms. The two trees are visited in parallel. At a given level of the recursion, two nodes are compared. If their boxes don't intersect, then the associated sectors don't intersect. The recursion stops. If the boxes intersect and are thicker than a given threshold then the thickest node is subdivided (to ensure that boxes of equivalent sizes are always compared, see [1]), and the recursion proceeds. In the remaining case, the nodes are not subdivided. They are leaves, and the associated sectors are then compared directly. If they intersect, the two nodes and all their ancestors are marked as intersecting. During the execution of the algorithm, for each node, a list of all nodes intersecting it is maintained.

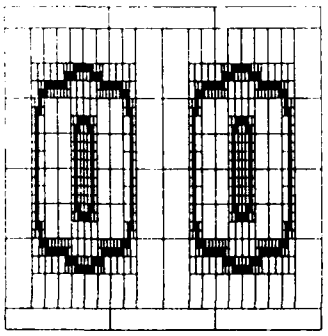


Figure 7. The Box Tree of the polynomial SHGC of Figure 6 after the computation of the intersection curves. The black squares are intersecting leaves.

We now give a procedural version of the algorithm. We assume we have the following functions:

Boxes-Intersect(N_1, N_2) returns true iff the boxes associated to the nodes N_1 and N_2 intersect. *Sectors-Intersect* (N_1, N_2) returns true iff the sectors associated to the nodes N_1 and N_2 intersect. This function in particular depends on the implementation: it may test directly planar approximations to the actual surfaces (as in the current implementation), or use a numerical scheme to compute the actual surface intersection between the sectors. The procedure *Subdivide*(N) builds the sons of the node N after having checked that these sons don't already exist. Notice that all these functions can be executed in essentially constant time.

```
Procedure Find-Box-Trees-Intersecting-Nodes (N1, N2);
begin
if Boxes-Intersect (N1, N2) then
  begin
  if (N1 ↑ .thickness < ε) and (N2 ↑ .thickness < ε)
    and Sectors-Intersect (N1, N2)
    then begin
      N1 ↑ .Mark ← Inter; N2 ↑ .Mark ← Inter;
      N1 ↑ .Intersecting-Nodes
          ← N1 ↑ .Intersecting-Nodes + N2;
      N2 ↑ .Intersecting-Nodes
          ← N2 ↑ .Intersecting-Nodes + N1
      end
    else (* The recursion proceeds *)
      if N1 ↑ .thickness > N2 ↑ .thickness
        then begin
          Subdivide(N1);
          for S1 in N1 ↑ .Sons do
            begin
            Find-Box-Trees-Intersecting-Nodes (S1, N2);
            if (S1 ↑ .Mark = Intersection)
              then N1 ↑ .Mark ← Intersection
            end
          end
        else begin
          Subdivide(N2);
          for S2 in N2 ↑ .Sons do
            begin
            Find-Box-Trees-Intersecting-Nodes (N1, S2);
            if (S2 ↑ .Mark = Intersection)
              then N2 ↑ .Mark ← Intersection
            end
          end
  end
end;
```

Figure 6 shows an example of the application of the algorithm to the computation of the intersection curves of a SHGC with a polynomial sweeping rule and a cylinder. Figure 7 shows the Box Tree associated to the polynomial SHGC, and Figure 8 shows the set operations between these two objects. Finally Figure 9 shows an example of set operations between a SHGC with edges and two cylinders. We now use the analogy between the Box Tree and the Quadtree and the classical analysis of [8] for Quadtree algorithms to get the complexity of the set operation algorithm.

Hunter and Steiglitz [8] describe an algorithm for computing the Quadtree for a polygon, i.e. a Quadtree such that only nodes intersecting the boundary of a polygon have children, and all nodes are colored either *inside*, *out-*

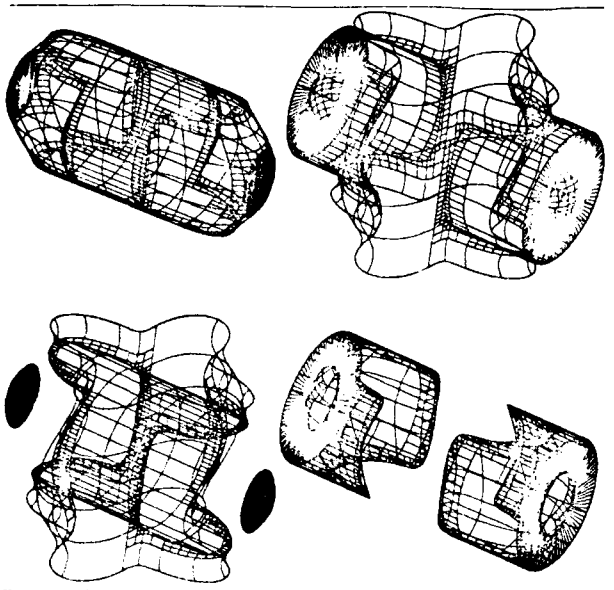


Figure 8. The set operations between a polynomial SHGC and a cylinder

side, or *intersection*. In our case, after having computed the intersection curves, the Box Tree can be condensed. In other terms, nodes all of whose children are non intersecting nodes can be marked as non intersecting (this additional step is required because two boxes can intersect although the corresponding sectors don't intersect, so two intersecting nodes may have only non intersecting children). After classification of the non intersecting nodes, *the Box Tree is then exactly equivalent in parameter space* to the Quadtree for polygon of the intersection curves.

The analysis of Hunter and Steiglitz for Quadtrees for polygons applies. They show that the total number of nodes in the tree is $O(p.2^q)$ where p is the perimeter of *the intersection curve(s)*, and q *is the depth of the tree*. In fact, here p is the perimeter in parameter space. For well behaved surfaces, the perimeter in world space is proportional to the perimeter in parameter space, and the complexity in world space stays the same. The actual number of nodes built during the set operations algorithm may in fact be slightly higher, as some nodes have to be condensed. Figure 10 shows on an example that the total number of nodes is in fact approximatively a linear function of 2^q, as predicted. The total number of intersecting sectors is bounded by the total number of nodes. This implies that S_2 and S_3 are both $O(p.2^q)$.

The number of nodes in each tree gives a lower bound to the complexity of the set operation algorithm. Once again, we can use Hunter's and Steiglitz's analysis to get an idea of the complexity of the localization step. They show that the subdivision of the intersecting nodes (analogous to our localization step) also has a $O(p.2^q)$ complexity. The localization step can be slightly more expensive than the subdivision step for Quadtrees, as a node may intersect several other nodes, and some intersecting boxes

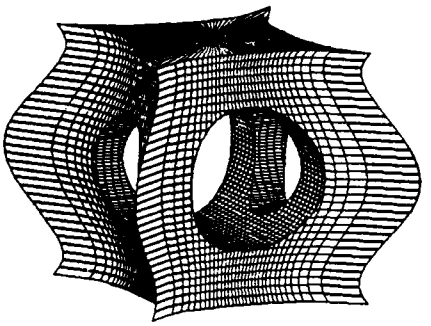


Figure 9. An example of set operations between a SHGC with edges and cylinders. The resulting solid is displayed by using a z-buffer algorithm to render an approximation of the sectors by planar faces.

don't actually correspond to sectors' intersections. If we assume most of them do, S_1 is $O(p.2^q)$. Figure 11 shows on an example that the number of intersections follows approximatively this law.

Our analysis shows that the total complexity of the set operation algorithms is approximatively $O(p.2^q)$. Notice that if we had used an homogeneous decomposition of the surface, the total number of sectors per object would have been $O(4^q)$, so the complexity of our algorithm is $O(\sqrt{n})$, where n is the equivalent number of faces for an homogeneous decomposition. The computing time on a lisp machine is 2 minutes for the complete set operation algorithm. Notice that the number of intersections at the deepest level of the tree is only 9301, although the total number of faces for each tree would be more than 25000 if a homogeneous decomposition was used.

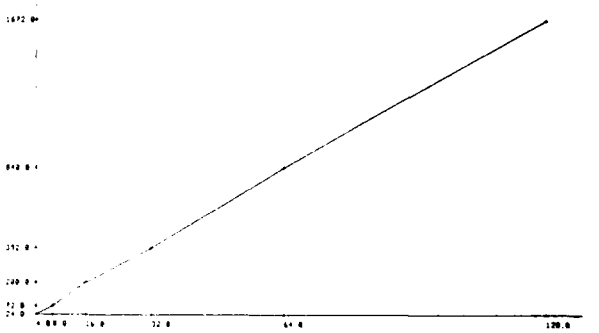


Figure 10. The total number of nodes in the Box Tree of the polynomial SHGC of Figure 6, drawn as a function of 2^q

4. The ray tracing algorithm

The basic algorithm is analogous to the algorithms for fractals [10] and Prism Trees [18]: at a given level, test the ray against a node. If it does not intersect its box, the recursion stops. Otherwise, the recursion proceeds. Test the ray against the fours sons of the node, and update a sorted active list of intersecting nodes. At the leaf level, compare directly the sectors of the active list to the ray,

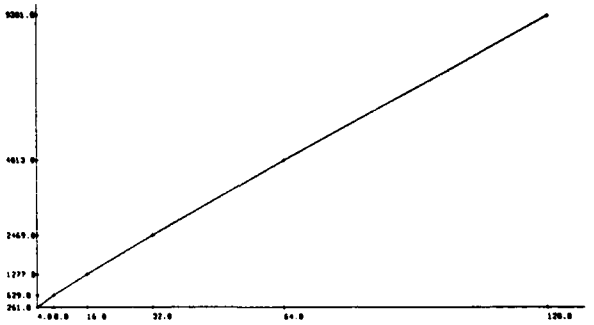


Figure 11. The number of intersection tests for the set operations between the SHGC and the cylinder of Figure 6, drawn as a function of 2^q.

the visible point corresponds to the first intersection. In fact this algorithm can be improved by observing that, as the boxes associated to the sons of a node are convex and disjoint, they can always be sorted from front to back along the ray. The trick of the algorithm is to explore the tree in a depth first manner, always visiting the sons of a node from front to back. This way, the first intersection found is guaranteed to be the good one. No active list is needed, and in the ideal case only one branch of the tree is visited. Again, we give a procedural version of the algorithm. We assume we have the following functions: *Sort-Intersections*(*Ray*, *N*) computes the intersections of the ray *Ray* with the boxes associated to the sons of *N* and returns the list of those intersections, sorted in increasing depth order. *Sector-Raycast*(*Ray*, *N*) returns the intersection of *Ray* and the sector associated to *N* (or nil if no such intersection exists). Again, this function may either compute the intersection of the ray and an approximating planar face, or compute the actual intersection. In the current implementation, it computes the intersection with the approximation, but the normal at the point is interpolated to yield a more precise value of the intensity.

```
Function Box-Tree-Raycast (Ray, Node) : point;
begin
Intersection-Point ← nil;
if Node ↑ .thickness < ε
  then (*Test ray against sector. Stop recursion*)
  Intersection-Point ← Sector-Raycast (Ray, Node)
  else begin (*Proceed. Test and sort sons.*)
    Subdivide(Node);
    Sorted-Sons-List
        ← Sort-Intersections (Ray, Node ↑ .Sons);
    Head-List ← Sorted-Sons-List;
    while (Head-List ≠ nil)and(not Intersection-Point)
      do begin
        Intersection-Point
          ← Box-Tree-Raycast(Ray, Head-List ↑ .Node);
        Head-List ← Head-List ↑ .next
        end;
    end;
Box-Tree-Raycast ← Intersection-Point;
end;
```

Figure 12. The ray traced image of a scene made of SHGC's.

Let us again try to analyze the complexity of the algorithm. In the ideal case the first branch of the tree visited leads to an intersecting sector. The complexity in this case is proportional to the depth q of the tree for each ray. Equivalently, it is $O(\log n)$ where n is the number of faces of the equivalent homogeneous decomposition. This time, the complexity of the construction of the tree is different, as it is likely that all the surface of the SHGC has to be subdivided up to the same level. This leads to a linear complexity in the number of faces, but the tree is computed just one time for all the image rendering.

Figure 12 shows the shaded ray traced image of a scene made of SHGC's. Notice the shadows. This image has been computed at a 512 × 512 resolution, with anti aliasing at the boundary between objects (using a double resolution there). The maximum depth of the tree is 6, so the number of equivalent polygons is around 40000. The computing time on a lisp machine is 3 hours. Figure 13 shows the average number of intersection tests per ray traced as a function of resolution for the example of Figure 12. In this case, the shadows are not computed. The figure shows that the number of intersections is roughly proportional to the depth of the tree, as predicted by the analysis.

Finally, Figure 3 shows the limbs and intersection curves of the union of a polynomial SHGC and a cylinder, rendered by using ray tracing at the limbs and intersection curves only. The limbs are computed by using the method described in [16]. The advantage of this method is its efficiency: the ray tracing is computed only on a few curves, instead of all the image pixels. It is particularly suited for line drawings, with a CPU time of about 5 minutes on a lisp machine.

Conclusion

We have presented the Box Tree, a new hierarchical representation for straight homogeneous generalized cylinders and have used it to efficiently compute set operations and ray tracing. The bulk of this paper was dedicated to graphics problems. The modelling system presented here, however, has been developped in the context of vision. It will be used to predict observable features in the images

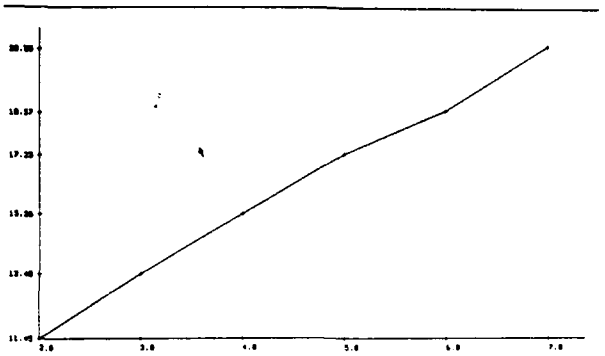


Figure 13. The average number of intersection tests per ray for the scene of Figure 12, drawn as a function of q.

of the modelled objects. For example we have shown elsewhere [16],[17] how the model of a SHGC could be used to predict its ribbons and find its axis in real images. Moreover, when several primitives are combined through set operations, the intersection curves between the surfaces of the primitives generate observable intensity discontinuities in images. The set operation algorithm described in Section 3 can be used to compute these curves and predict the properties of their projections in images. Finally, we believe that a wider class of generalized cylinders and joining operations are necessary to model classes of real objects. Our next step will be to investigate the representation of other primitives (for example generalized cylinders whose spine is a space curve, or whose cross section is not orthogonal to the axis, see [17]) and joining operations (like blended surfaces and articulations).

References

[1] Ballard, D.H., Strip Trees: A hierarchical representation for curves. Comm. of the ACM, Vol 24, No 5 (May 1981), pp. 310-321.

[2] Binford, T.O., Visual perception by computer, Proc. IEEE Conf. on Systems and Control, Miami (December 1971).

[3] Brooks, R.A., Symbolic reasoning among 3D models and 2D images, Artificial Intelligence 17 (1981), pp. 285-348.

[4] Carlson, W.E., An algorithm and data structure for 3D object synthesis using surface patch intersections, Comp. Gr. SIGGRAPH 82 Conf. Proc. 16 (1982), 3, pp.255-264.

[5] Catmull, E., A subdivision algorithm for computer display of curved surfaces, University of Utah, Salt Lake City, 1974.

[6] Fournier, A., Russel, D., and Carpenter, L., Computer rendering of stochastic models, Comm. of the ACM, Vol 25, No 6 (June 1982), pp. 371-384.

[7] Hoffmann, C., and Hopcroft, J., "Automatic surface generation in Computer Aided Design", TR 85-661, Dept. of Comp. Science, Cornell University, 1985.

[8] Hunter, G.M., and Steiglitz, K., Operations on images using quad trees, IEEE Trans. Patt. Analysis and Machine Intell., Vol PAMI-1, No 2 (1979).

[9] Jackins,C.L., and Tanimoto,S.L., Oct-trees and their use in representing 3D objects, Dept. of Comp. Science, *Univ. of Washington*, Seattle, Techn. Rep. 79-07-06 (1980).

[10] Kajiya, J.T., New techniques for ray tracing procedurally defined objects, Proc. SIGGRAPH 83 (july 1983), pp. 91-102.

[11] Kalay, Y.E., "Determining the spatial containment of a point in general polyhedra", Computer Vision, Graphics, and Image Processing 19, pp. 303-334, 1982.

[12] Mantyla, M., and Tamminen, M. ,Localized set operations for solid modelling, Proc. SIGGRAPH 83 (july 1983), pp. 279-288.

[13] Marr,D., and Nishihara,K., Representation and recognition of the spatial organization of three dimensional shapes, Proc. Royal Soc. of London, B-200, pp. 269-294 (1977).

[14] Meagher, D., Geometric modelling using octree encoding, Comp. Gr. Image Pr., Vol 19 (1982), pp. 129-147.

[15] Nevatia, R., **Machine perception**, Prentice-Hall, Englewood Cliffs, NJ, 1982.

[16] Ponce, J., and Chelberg, D., Finding the limbs and cusps of generalized cylinders, Proc. of the 4^{th} IEEE International Conference on Robotics and Automation, April 1987.

[17] Ponce, J., Chelberg, D., and Mann, W., Invariant properties of the projections of straight homogeneous generalized cylinders, submitted to to the First International Conference on Computer Vision, 1987.

[18] Ponce,J., and Faugeras, O., An object centered hierarchical representation for 3D objects: the Prism Tree, Comp. Vis. Gr. Image Pr., 1987 (to appear).

[19] Requicha, A.A.G., and Voelcker, H.B., Solid modeling: a historical summary and contemporary assessment, IEEE Comp. Graphics and Appl., Vol 2, No 2 (1982), pp. 9-24

[20] Roth, S.D., Ray Casting for modelling solids, Comp. Gr. Image Pr., Vol 18 (1982), pp. 109-144.

[21] Samet, H., Neighbor finding techniques for images represented by Quadtrees, Comp. Gr. Image Pr., Vol 18 (1982).

[22] Samet, H., The Quadtree and related hierarchical data structures, Computing Surveys, Vol. 16, No 2, pp. 187-260 (1984).

[23] Shafer, S.A., and Kanade, T., The theory of straight homogeneous generalized cylinders, Carnegie Mellon Univ., Comp. Sc. Dept. Tech. Rep. CMU-CS083-105 (1983).

[24] Tanimoto, S., and Pavlidis, T., A hierarchical data structure for picture processing, Comp. Gr. Image Pr., Vol 4, No 2 (1975).

[25] Tilove, R.B., Set membership classification: a unified approach to geometric intersections problems, IEEE Trans. on Comp., C 29 (10) (1980), pp. 874-883.

[26] Whitted, T., An improved illumination model for shaded display, CACM, Vol. 23, No 6 (1980), pp. 343-349.

Line-Drawing Interpretation : Straight-Lines and Conic-Sections

Vic Nalwa

A.I. Lab., Stanford University, CA 94305

Abstract

Line-drawings corresponding to orthographically projected images of man-made scenes often exhibit instances of straight-lines and conic-sections, i.e., ellipses, parabolas, and hyperbolas. We investigate constraints imposed on the scene by such instances under the assumption of general *viewpoint, i.e., under the assumption that the properties of the mapping of the viewed surface onto the line-drawing are stable under perturbation of the viewpoint within some open set on the Gaussian sphere. The viewed surfaces are assumed to be piecewise C^3. The main results of this paper are that straight-lines, ellipses, parabolas, and hyperbolas in line-drawings are orthographic projections of scene-edges which are also straight-lines, ellipses, parabolas, and hyperbolas, respectively. Further, continuous-surface-tangent depth discontinuities which orthographically project onto straight-lines can be described locally by developable surfaces, and those which orthographically project onto conic-sections can be described locally by quadric surfaces. It is also shown that scene events which orthographically project onto straight-lines or conic-sections cannot be combinations of viewpoint-independent and viewpoint-dependent edges.*

I. Introduction

Line-drawings are representations of edges in an image. An edge in an image may be caused by various events in the scene. The scene-event may be a surface-tangent discontinuity, a depth discontinuity, a surface-reflectance discontinuity or an illumination discontinuity (shadow). Fig. 1 illustrates the various cases. Notice that a depth discontinuity may also simultaneously be a surface-tangent discontinuity. We distinguish this from a continuous-surface-tangent depth discontinuity. While surface-tangent discontinuities, illumination discontinuities, and surface-reflectance discontinuities constitute viewpoint-independent scene-edges, continuous-surface-tangent depth discontinuities are viewpoint-dependent.

Although the depth information is lost under projection, we can nevertheless impose some constraints on the 3-D scene by investigating the 2-D configuration of the corresponding line-drawing. We restrict our attention to orthographic projection here. Orthographic projection is a reasonable approximation to perspective projection whenever the angle subtended by the object at the viewpoint is small. Orthographically projected line-drawings of man-made scenes often exhibit instances of straight-lines and conic-sections, i.e., ellipses, parabolas, and hyperbolas. We investigate constraints imposed on the scene by such instances under the assumption of *general* viewpoint, i.e., under the assumption that the mapping of the viewed surface onto the line-drawing is stable under perturbation of the viewpoint within some open set on the Gaussian sphere[1]. The stability we demand includes the requirement that the categorization of the curve in the line-drawing as a straight-line, ellipse, parabola, or hyperbola, be preserved under perturbation of viewpoint. Other properties of the mapping with respect to which we require stability will be mentioned as we proceed. As long as the total number

[1] The Gaussian sphere is a device to represent orientation in space. Each point on a unit sphere, called the Gaussian sphere (see [Hilbert and Cohn-Vossen, 1952]), corresponds to the unit vector from the center of the sphere to that point.

of such requirements is finite, the set of viewing directions on the Gaussian sphere where the *general* viewpoint assumption is invalid is closed and has measure zero[2]. It should be noted that given the loss of information about the third dimension under projection, one cannot hope to make any substantial deductions about the imaged surfaces without the *general* viewpoint assumption.

It is easy to see that the only space curves which orthographically project, under *general* viewpoint, to straight-lines, ellipses, parabolas, or hyperbolas are those which belong to the corresponding family. This observation, however, pertains only to all viewpoint-independent scene-edges. The situation is more subtle for continuous-surface-tangent depth discontinuities where the viewing direction is tangential to the surface. To illustrate this, we present in Fig. 2 an instance of a C^1 curve in a line-drawing which is the orthographic projection of a C^0 curve in space. The surface in the figure consists of a hemisphere smoothly joined to one end of a right circular cylinder.

We examine lines resulting from depth discontinuities here. All viewpoint-independent edges can be included in the following analysis by assuming, without any loss of generality, all non-depth discontinuities to be replaced by discontinuous-surface-tangent depth discontinuities. Such a replacement retains the viewpoint-independent character of the scene-edges while making them amenable to analysis as depth discontinuities. It will turn out that combinations of viewpoint-independent and viewpoint-dependent scene-edges cannot orthographically project to straight-lines or conic-sections under *general* viewpoint. Pairs of straight-lines are exceptions to this rule. These are degenerate conic-sections which may have one of the two lines viewpoint-independent while the other is viewpoint-dependent. Throughout our discussion we will treat pairs of straight-lines to be instances of straight-lines rather than of conics.

II. Preview of Results

We summarize here the results of this paper. It is assumed that the projected surfaces are piecewise C^3, i.e., the surfaces comprise finitely many C^3 patches, each bounded by a finite piecewise C^3 curve. This restriction does not significantly constrain the domain as long as we do not require a priori knowledge of the C^3 surface-patch boundaries. The other assumptions are explicitly stated in the theorems. The straight-lines and conic-sections in the line-drawing need not be continuous for the theorems to be valid, i.e., the constraints imposed on the viewed surface hold even if the curves in the line-drawing are fragmented.

Theorem 1 : *Straight-lines in a line-drawing obtained under orthographic projection with a* general *viewpoint are projections of straight-lines in space.*

Theorem 2 : *Ellipses, parabolas and hyperbolas in a line-drawing obtained under orthographic projection with a* general *viewpoint are projections of ellipses, parabolas and hyperbolas, respectively, in space.*

Theorem 3 : *Circles in a line-drawing obtained under orthographic projection with a* general *viewpoint are projections of circles in space which are confined to lie in planes parallel to the image plane.*

Theorem 4 : *Scene events which orthographically project, under* general *viewpoint, onto straight-lines or conic-sections in line-drawings must either be completely viewpoint-independent or completely viewpoint-dependent.*

Theorem 5 : *Continuous-surface-tangent depth discontinuities which orthographically project, under* general *viewpoint, onto straight-lines in line-*

[2] An arbitrary set in any manifold has measure zero if its preimage, for every local parametrization, can be covered by a countable number of rectangular solids with an arbitrarily small total volume in Euclidean space. For our purposes it suffices to note that the Gaussian sphere is a two dimensional manifold and consequently all sets of points and lines on it have measure zero. The null set of course has measure zero. On the other hand, no open set on the Gaussian sphere has measure zero.

It can be shown that the union of a countable number of sets of measure zero is also of measure zero (see [Guillemin and Pollack, 1974]). Therefore, the union of a finite number of closed sets of measure zero is closed and of measure zero. The complement of such a union is open and of full measure.

drawings can be described locally by developable surfaces.

Theorem 6 : *Continuous-surface-tangent depth discontinuities which orthographically project, under* general *viewpoint, onto conic-sections in line-drawings can be described locally by quadric surfaces. Specifically, continuous-surface-tangent depth discontinuities projecting onto ellipses by ellipsoids or hyperboloids of one sheet, onto parabolas by elliptic paraboloids or hyperbolic paraboloids, and onto hyperbolas by hyperboloids of one sheet or hyperboloids of two sheets. Furthermore, these quadric surfaces are completely determined, up to three degrees of freedom, by the conic-sections in the line-drawings.*

Theorem 7 : *Depth discontinuities which orthographically project, under* general *viewpoint, onto circles in line-drawings can be described locally by spheres whose radii are identical to the radii of the circles in the line-drawings.*

Section III provides a formulation for the geometry of the problem. The results derived within are used throughout the rest of the paper. Sections IV, V and VI discuss straight-lines, conic-sections, and circles, respectively. In section VII we extend the results derived in the preceding sections. We conclude with section VIII.

III. Preliminaries

We begin by formulating the geometry of the problem before we get down to the theorems and their proofs. The viewed surface is assumed to be piecewise C^3. A piecewise C^3 surface is defined to comprise finitely many C^3 patches, each bounded by a finite piecewise C^3 curve.

Consider the projection geometry shown in Fig. 3. The projection direction is along the z-axis. The figure also shows a curve which results from the intersection of the viewed surface with a plane parallel to the y-z plane. The depth discontinuity, say (x_i, y_i, z_i), which lies on this curve has its tangent oriented along the direction of projection. If we perturb the viewpoint by the counter-clockwise angle Δ about the x-axis, it is clear that the depth discontinuity before perturbation of viewpoint may no longer remain a depth discontinuity. Fig. 4 shows the original depth discontinuity (y_i, z_i) and the new depth discontinuity (y_i', z_i') which lies on the same cross-sectional curve. Note that the new discontinuity is expressed in the perturbed frame of reference which has its z-axis oriented along the new projection direction.

Let the curvature of the cross-sectional curve at (y_i, z_i) be κ_{xi}, where the subscript x refers to the line in the x-y plane about which the viewpoint is perturbed, i.e., the x-axis. We need to interpret κ_{xi}. As shown in Fig. 3, the surface cross-section normal, in general, forms a non-zero angle with the surface normal at the same point. Let us call this angle ϕ_{xi}. The subscript x on ϕ indicates that it is a function of the axis about which the viewpoint is perturbed. Then, we may write $\kappa_{xi} = \kappa_i / \cos\phi_{xi}$ (see [Lipschutz, 1969]) where κ_i is the normal curvature of the surface in the projection direction at the point (x_i, y_i, z_i). The radius of curvature of the cross-sectional curve at the depth discontinuity is $r_{xi} = \frac{1}{\kappa_{xi}}$ and so we have the equivalent relation $r_{xi} = r_i \cos\phi_{xi}$ where $r_i = \frac{1}{\kappa_i}$ is the radius of normal curvature in the projection direction.

In case of a tangent discontinuity on the cross-sectional curve at (y_i, z_i), we may let $\kappa_{xi} \to \infty$ and all the following analysis will remain valid. If the cross-sectional curve has a curvature discontinuity at (y_i, z_i), then we assume κ_{xi} to take on one of the two limiting values depending on which side of the original depth discontinuity the perturbed depth discontinuity lies[3]. This remark will be further clarified as we proceed. It will be shown later that for orthographic projections which are conic-sections under *general* viewpoint, curvature discontinuities are disallowed at the depth discontinuities. For straight-lines, on the other

[3] It is implicitly assumed here that the projection direction is not tangential to the boundary of a C^3 surface patch at any depth discontinuity. Observe that the set of orientations which are tangential to any boundary is closed and of measure zero on the Gaussian sphere. As the number of boundaries is finite, the union of such sets is also closed and of measure zero. Hence, the assumption is valid under *general* viewpoint.

hand, it will be shown that we must either have curvature discontinuities at the depth discontinuities on every cross-sectional curve or at none. However, even here the viewpoint will turn out to be non-*general* in that the set of such viewpoints is closed and of measure zero.

It is reasonable to assume that $\kappa_{xi} \neq 0$. The justification for this assumption will be presented below. Then, ignoring higher order terms, we could describe the local cross-sectional curve about (y_i, z_i) as follows.

$$(y - y_i) = \tfrac{1}{2}\kappa_{xi}(z - z_i)^2 \qquad (1)$$

Perturbation of the reference frame causes the following transformation of coordinates.

$$x \rightarrow x'$$
$$y \rightarrow y' \cos\Delta - z' \sin\Delta$$
$$z \rightarrow y' \sin\Delta + z' \cos\Delta$$

Equation (1) will then read

$$(y' \cos\Delta - z' \sin\Delta - y_i) = \tfrac{1}{2}\kappa_{xi}(y' \sin\Delta + z' \cos\Delta - z_i)^2 . \qquad (2)$$

The higher order terms may be tentatively neglected because their contribution becomes relatively insignificant as the size of the perturbation is reduced. However, it may turn out that we are unable to satisfy our requirements by only constraining the local tangents and curvatures. In that case we would have to include the higher order terms in our analysis.

The new depth discontinuities under perturbation of viewpoint can be found by equating (dy'/dz') in equation (2) to zero. This leads to equation (3) where (y_i', z_i') are the coordinates of the new depth discontinuity corresponding to the original depth discontinuity with the same x-coordinate. Equation (4) is obtained by substituting for the equality (3) in equation (2).

$$y_i' \sin\Delta + z_i' \cos\Delta = z_i - \frac{1}{\kappa_{xi}}\tan\Delta \qquad (3)$$

$$y_i' \cos\Delta - z_i' \sin\Delta = y_i + \frac{1}{2\kappa_{xi}}\tan^2\Delta \qquad (4)$$

Using (3) and (4), and making the substitution $r_{xi} = \frac{1}{\kappa_{xi}}$, we finally express the new depth discontinuities as follows.

$$x_i' = x_i \qquad (5)$$
$$y_i' = (y_i + z_i \tan\Delta - \tfrac{1}{2} r_{xi} \tan^2\Delta)\cos\Delta$$
$$z_i' = (z_i - y_i \tan\Delta - r_{xi} \tan\Delta - \tfrac{1}{2} r_{xi} \tan^3\Delta)\cos\Delta$$

Before proceeding to the next section, we present a justification for the assumption that $\kappa_{xi} \neq 0$. The reader may wish to postpone going through this argument till later. First, it can be argued that κ_{xi} is either zero all along the depth discontinuity curve or it is not zero anywhere along it. The reason for this is that, under perturbation of viewpoint, points with non-zero κ_{xi} move an order of magnitude slower than points with zero κ_{xi}. Hence, if κ_{xi} were not uniformly zero or non-zero along the depth discontinuity curve, then the projection would not remain a straight-line or conic-section under perturbation of viewpoint. This argument could be made more formal only after the presentation of the following two sections. Next, we argue that κ_{xi} cannot be zero all along the depth discontinuity curve, for a *general* viewpoint. If any of the cross-sectional curves has a tangent discontinuity at the depth discontinuity, then κ_{xi} will be non-zero there. Let us assume that this is not so, i.e., κ_{xi} is zero all along the depth discontinuity curve and the surface-tangent is continuous along this curve. Then, no depth discontinuity on any cross-sectional curve will remain a depth discontinuity under perturbation of viewpoint. It follows that, under *general* viewpoint, there must exist open intervals on the depth discontinuity curve which lie within the interior of C^3 patches. Now, every non-boundary depth discontinuity point belongs to one of the following categories : elliptic, parabolic, hyperbolic or planar (see [Lipschutz, 1969]). Because every planar depth discontinuity must transit to a non-planar depth discontinuity under perturbation of viewpoint[4], we will always

[4] Planar points form closed sets on the surface. A great circle on the Gaussian sphere describes the viewing directions for which points in any one such set may lie on a depth discontinuity curve. As every great circle on the Gaussian sphere is a closed set of measure zero, the described transition must occur under perturbation of the viewpoint.

have some open interval on the depth discontinuity curve which lies within some open surface set consisting solely of elliptic, parabolic, or hyperbolic points. In case one such elliptic patch exists, κ_i, and therefore κ_{xi}, cannot be identically zero along the depth discontinuity curve. In case of a parabolic patch, each point has only one direction in which the surface curvature, κ_i, is zero. Consider the cross-sectional curve, of the type shown in Fig. 3, on which one such point lies. Because our parabolic surface patch is C^3, the asymptotic directions along the cross-sectional curve must either map onto a point or a C^1 curve on the Gaussian sphere. In either case, κ_i along the cross-sectional curve can be zero only for a closed set of measure zero on the Gaussian sphere. Therefore, κ_{xi} cannot be identically zero along the depth discontinuity curve, under *general* viewpoint. An identical argument holds for hyperbolic points, except that now we consider two separate mappings, one for each of the asymptotic directions. In this case, the desired result follows from the observation that the union of two closed sets of measure zero is also closed and of measure zero. This completes the argument that for a piecewise C^3 surface and a *general* viewpoint, κ_{xi} is not zero at any point on the depth discontinuity curve.

IV. Straight-Lines

Theorem 1 : *Straight-lines in a line-drawing obtained under orthographic projection with a* general *viewpoint are projections of straight-lines in space.*

Proof : For reasons discussed in the introduction, it is sufficient to verify the truth of the theorem for depth discontinuities. Consider any three points, say (x_1, y_1, z_1), (x_2, y_2, z_2) and (x_3, y_3, z_3), which project to a straight-line in a line-drawing. Collinearity of the orthographic projection of the three points, under perturbation of viewpoint, requires that

$$\frac{y_1' - y_2'}{x_1' - x_2'} = \frac{y_1' - y_3'}{x_1' - x_3'}. \tag{6}$$

Substitution, using (5), and simplification using the collinearity of (x_1, y_1), (x_2, y_2) and (x_3, y_3), the original orthographic projections of the three points, leads to the following equality.

$$\left(\frac{z_1 - z_2}{x_1 - x_2} - \frac{z_1 - z_3}{x_1 - x_3}\right)\frac{2}{\tan\Delta} = \frac{r_{x1} - r_{x2}}{x_1 - x_2} - \frac{r_{x1} - r_{x3}}{x_1 - x_3} \tag{7}$$

If the three points are collinear in space, then the left-hand-side of the equation is identically zero and we have the following constraint on the radii of curvature, r_{xi}, at the points along the cross-sectional curves perpendicular to the axis about which the viewpoint is perturbed, i.e., the x-axis here.

$$\frac{r_{x1} - r_{x2}}{r_{x1} - r_{x3}} = \frac{x_1 - x_2}{x_1 - x_3} \tag{8}$$

If the three points are not collinear, then the left-hand-side of (7) is a function of the angle Δ while the right-hand-side is constant for any choice of points and curvatures. Hence, the equation cannot be satisfied under such circumstances.

Q.E.D.

We now consider the constraints imposed upon the viewed surface by the equality (8). We use the previously discussed relation $r_{xi} = r_i \cos\phi_{xi}$ for this purpose. Although ϕ_{xi}, in general, depends on the point under consideration, for a straight-line projection under *general* viewpoint it is constant because of Theorem 1 and may be expressed as ϕ_x. Consequently, equation (8) can be rewritten as follows.

$$\frac{r_1 - r_2}{r_1 - r_3} = \frac{x_1 - x_2}{x_1 - x_3} \tag{9}$$

This implies that the radius of normal curvature, in the direction of projection, along the straight surface depth discontinuity varies linearly with the distance measured along the line. Hence, it is either identically zero or it is zero at most one isolated point. In the former case the edge is a tangent discontinuity which is viewpoint-independent while in the latter case it is a viewpoint-dependent depth discontinuity. Henceforth in this discussion, we will restrict ourselves to the latter case.

From the linear variation of r_i it should be

clear that either the curvatures of the cross-sectional curves are discontinuous at every depth discontinuity or at none. We may assume, without any loss of generality, that the curvatures are continuous at every depth discontinuity because the set of viewpoints for which this assumption is false is closed and of measure zero. Now we want to deduce that the surface is locally C^2. The surface would not be locally C^2 only if some depth discontinuity point is on the boundary of a C^3 surface patch and it does not have a twice differentiable neighborhood. For every such point, as previously shown, the tangent to the boundary does not lie along the projection direction, under *general* viewpoint. Then, if any such point is to lie on the depth discontinuity curve, the cross-sectional curve through it cannot be C^2. Hence, the surface must be locally C^2.

It is not hard to see that the normal curvature of the surface in the direction of the depth discontinuity curve is one of the principal curvatures and is identically zero. The other principal curvature has a fixed orientation, orthogonal to the straight-line depth discontinuity curve, and varies linearly along it. As these surface properties are preserved under perturbation of viewpoint, the surface must be locally developable, i.e., cylindrical, conical, or tangential developable (see [Hilbert and Cohn-Vossen, 1952]).

V. Conic-Sections

Theorem 2 : *Ellipses, parabolas and hyperbolas in a line-drawing obtained under orthographic projection with a* general *viewpoint are projections of ellipses, parabolas and hyperbolas, respectively, in space.*

Proof : For reasons discussed in the introduction, it is sufficient to verify the truth of the theorem for depth discontinuities. We first show that space curves which project to conics under *general* viewpoint must be planar. Straight-lines are excluded from this discussion.

Consider any five points, say (x_i, y_i, z_i), $i = 1..5$, which project onto the conic. The orthographic projection direction, as before, is the z-axis. The five chosen points uniquely determine the conic, $ax^2 + bxy + cy^2 + dx + ey + f = 0$. The conic coefficients are the solution to $\mathbf{P}\,[a\ b\ c\ d\ e\ f\ ]^T = 0$, where

$$\mathbf{P} = \begin{bmatrix} x_1^2 & x_1y_1 & y_1^2 & x_1 & y_1 & 1 \\ x_2^2 & x_2y_2 & y_2^2 & x_2 & y_2 & 1 \\ x_3^2 & x_3y_3 & y_3^2 & x_3 & y_3 & 1 \\ x_4^2 & x_4y_4 & y_4^2 & x_4 & y_4 & 1 \\ x_5^2 & x_5y_5 & y_5^2 & x_5 & y_5 & 1 \end{bmatrix}.$$

Now let us choose a sixth point, say (x_6, y_6, z_6), which also projects onto the conic determined above. Then, we may write

$$[x_6^2\ \ x_6y_6\ \ y_6^2\ \ x_6\ \ y_6\ \ 1] = [\eta_1\ \eta_2\ \eta_3\ \eta_4\ \eta_5]\,\mathbf{P}\,. \qquad (10)$$

The η_i are uniquely determined owing to the linear independence of the rows of $\mathbf{P}$. Using (10), we may express the η_i as follows.

$$[\eta_1\ \eta_2\ \eta_3\ \eta_4\ \eta_5] = [x_6^2\ \ x_6y_6\ \ y_6^2\ \ x_6\ \ y_6\ \ 1]\,\mathbf{P}^T(\mathbf{P}\mathbf{P}^T)^{-1} \qquad (11)$$

Now, we perturb the viewpoint about the x-axis by the counter-clockwise angle Δ, and once again use the expressions (5) for the new depth discontinuities. For small Δ, we may substitute $\tan\Delta$ by Δ and $\cos\Delta$ by unity. Then, the vector $[x_i^2\ \ x_i\,y_i\ \ y_i^2\ \ x_i\ \ y_i\ \ 1]$ transforms to

$$[x_i^2\ \ x_i(y_i + z_i\Delta - \tfrac{1}{2}r_{xi}\Delta^2)\ \ (\,.\,)^2\ \ x_i\ \ (\,.\,)\ \ 1].$$

$\mathbf{P}$ is correspondingly transformed to $\mathbf{P}'$. For the projected curve to remain a conic, it is necessary that

$$[x_6^2\ \ x_6(y_6 + z_6\Delta - \tfrac{1}{2}r_{x6}\Delta^2)\ \ (\,.\,)^2\ \ x_6\ \ (\,.\,)\ \ 1] = [\lambda_1\ \lambda_2\ \lambda_3\ \lambda_4\ \lambda_5]\,\mathbf{P}' \qquad (12)$$

for some λ_i. Ignoring higher order terms, we may approximate λ_i by $(\eta_i + p_i\Delta + q_i\Delta^2)$. Then, equating the coefficients of the Δ terms, we get the following equation.

$$[\eta_1\ \eta_2\ \eta_3\ \eta_4\ \eta_5]\,\mathbf{Q} + [p_1\ p_2\ p_3\ p_4\ p_5]\,\mathbf{P} = [0\ \ x_6z_6\ \ 2y_6z_6\ \ 0\ \ z_6\ \ 0] \qquad (13)$$

$$\textit{where} \quad \mathbf{Q} = \begin{bmatrix} 0 & x_1z_1 & 2y_1z_1 & 0 & z_1 & 0 \\ 0 & x_2z_2 & 2y_2z_2 & 0 & z_2 & 0 \\ 0 & x_3z_3 & 2y_3z_3 & 0 & z_3 & 0 \\ 0 & x_4z_4 & 2y_4z_4 & 0 & z_4 & 0 \\ 0 & x_5z_5 & 2y_5z_5 & 0 & z_5 & 0 \end{bmatrix}$$

We may rewrite (13), using (11), as follows.

$$[x_6^2 \;\; x_6y_6 \;\; y_6^2 \;\; x_6 \;\; y_6 \;\; 1]\,\mathbf{P}^T(\mathbf{P}\mathbf{P}^T)^{-1}\mathbf{Q} - [0 \;\; x_6z_6 \;\; 2y_6z_6 \;\; 0 \;\; z_6 \;\; 0] + [p_1 \;\; p_2 \;\; p_3 \;\; p_4 \;\; p_5]\,\mathbf{P} = 0 \qquad (14)$$

Now, notice that the first two row vectors have zeros for the first, fourth and sixth elements. Hence, we may rewrite (14) as

$$[x_6^2 \;\; x_6y_6 \;\; y_6^2 \;\; x_6 \;\; y_6 \;\; 1]\,\mathbf{P}^T(\mathbf{P}\mathbf{P}^T)^{-1}\mathbf{Q}' - [x_6 \;\; 2y_6 \;\; 1]z_6 + [p_1 \;\; p_2]\,\mathbf{R} = 0 \qquad (15)$$

$$\textit{where} \quad \mathbf{Q}' = \begin{bmatrix} x_1z_1 & 2y_1z_1 & z_1 \\ x_2z_2 & 2y_2z_2 & z_2 \\ x_3z_3 & 2y_3z_3 & z_3 \\ x_4z_4 & 2y_4z_4 & z_4 \\ x_5z_5 & 2y_5z_5 & z_5 \end{bmatrix}$$

and $\mathbf{R}$ *depends on* (x_i, y_i), $i = 1..5$.

Because $\mathbf{P}$ was a full rank matrix, the two rows of $\mathbf{R}$ are independent. It follows from this observation that the combination of the first two terms in equation (15) must lie in the plane defined by the two rows of $\mathbf{R}$, irrespective of our choice of (x_6, y_6, z_6), as long as this point projects onto the conic, i.e., x_6 and y_6 satisfy (10). Now, for a curve spanned by a row vector, say $\mathbf{v}(t)$, to be planar its torsion must be identically zero, i.e., $\det\left([\mathbf{v}'(t)\;\mathbf{v}''(t)\;\mathbf{v}'''(t)]^T\right)$ must equal zero (see [Lipschutz, 1969]). Hence, it is necessary that the combination of the first two terms in (15) satisfy this requirement, under variation of $(x_6, y_6, z_6) = (x(t), y(t), z(t))$. Note that the original coordinate frame could always be chosen so that the ellipses, parabolas, and hyperbolas under consideration are parameterized as $x(t) = m\cos t$ and $y(t) = M\sin t$, $x(t) = mt$ and $y(t) = Mt^2$, and $x(t) = m\cosh t$ and $y(t) = M\sinh t$, respectively. Substitution of the parametric forms of x_6, y_6, and z_6 into the zero torsion equation will lead to a third order ordinary differential equation in $z(t)$. This differential equation will be satisfied by $z(t) = ax(t) + by(t) + c$ because (15) is just a restatement of (13), and it can easily be verified that (13) is satisfied by this expression for $z(t)$ if we make all the p_i zero. As a, b and c, and therefore $z(t)$, are uniquely determined by $z(0)$, $z'(0)$, and $z''(0)$, it follows from the theory of ordinary differential equations that the planar expression for $z(t)$ is the unique solution on any interval containing $z(0)$ (see [Coddington, 1961]). To be thorough, one must also check that the Lipschitz condition is satisfied by the differential equation. This, however, is rarely a problem.

We have shown above that the space curves corresponding to conic orthographic projections under *general* viewpoint are restricted to be planar. If the orthographic projection of a planar curve is quadratic, then the original curve must also be a quadratic. In fact, the sign of the discriminant $(b^2 - 4ac)$ of the quadratic, $ax^2 + bxy + cy^2 + dx + ey + f = 0$, is unchanged under orthographic projection. These claims can easily be verified by noting that any orthographic projection of a planar curve may be computed, up to a rotation, by compressing the curve along some direction in the plane of the curve. Hence, we have proven that ellipses, parabolas, and hyperbolas in a line-drawing obtained under orthographic projection with a *general* viewpoint are projections of space curves which themselves are ellipses, parabolas, and hyperbolas, respectively.

Q.E.D.

As we did for straight lines previously, we now consider the constraints other than planarity which can be imposed on the viewed surface in the vicinity of the depth discontinuities. Once again consider the substitution of $(\eta_i + p_i\Delta + q_i\Delta^2)$ for λ_i in equation (12). Also note that we have shown above that all the p_i are zero and that $z_i = ax_i + by_i + c$. Using this information and equating the coefficients of the Δ^2 terms, we get the following equation.

$$[\eta_1 \;\; \eta_2 \;\; \eta_3 \;\; \eta_4 \;\; \eta_5]\,\mathbf{S} + [q_1 \;\; q_2 \;\; q_3 \;\; q_4 \;\; q_5]\,\mathbf{P} = [0 \;\; -\tfrac{1}{2}x_6r_{z6} \;\; -y_6r_{z6} \;\; 0 \;\; -\tfrac{1}{2}r_{z6} \;\; 0] \qquad (16)$$

where

$$\mathbf{S} = \begin{pmatrix} 0 & -\frac{1}{2}x_1 r_{x1} & -y_1 r_{x1} & 0 & -\frac{1}{2}r_{x1} & 0 \\ 0 & -\frac{1}{2}x_2 r_{x2} & -y_2 r_{x2} & 0 & -\frac{1}{2}r_{x2} & 0 \\ 0 & -\frac{1}{2}x_3 r_{x3} & -y_3 r_{x3} & 0 & -\frac{1}{2}r_{x3} & 0 \\ 0 & -\frac{1}{2}x_4 r_{x4} & -y_4 r_{x4} & 0 & -\frac{1}{2}r_{x4} & 0 \\ 0 & -\frac{1}{2}x_5 r_{x5} & -y_5 r_{x5} & 0 & -\frac{1}{2}r_{x5} & 0 \end{pmatrix}$$

But this is precisely of the same form as equation (13) with q_i replacing p_i and $-\frac{1}{2}r_{xi}$ replacing z_i. Hence, by the same arguments as above, we may conclude that the radius of curvature, r_{xi}, at the depth discontinuity (x_i, y_i, z_i), along the cross-sectional curve formed by the intersection of the viewed surface with a plane parallel to the y-z plane, can be expressed as $(\alpha x_i + \beta y_i + \gamma)$. Employing the relation $r_{xi} = r_i \cos\phi_{xi}$, we get

$$r_i \cos\phi_{xi} = (\alpha x_i + \beta y_i + \gamma).$$

First, note that ϕ_{xi} can be expressed as $(\phi_x + \phi_i)$ where ϕ_i is determined by the tangent to the conic-section in space and ϕ_x depends on the axis about which the viewpoint was perturbed, in this case the x axis. Next, observe that α, β and γ must depend on ϕ_x. Hence, we rewrite the expression as,

$$r_i = \frac{\alpha(\phi_x)\, x_i + \beta(\phi_x)\, y_i + \gamma(\phi_x)}{\cos(\phi_x)\cos(\phi_i) - \sin(\phi_x)\sin(\phi_i)}. \qquad (17)$$

In the discussion so far, we have considered the perturbation to be about the x-axis. However, we were free to perturb the viewpoint about any axis in the x-y plane. Because r_i is a surface property independent of the direction of perturbation, we require that the right-hand-side of (17) be independent of ϕ_x.

If the numerator in (17) is not identically zero, then it can be zero at most two points along the depth discontinuity curve. Hence, either r_i is identically zero or it is zero at most two isolated points. We conclude that no edge which projects to a conic-section under *general* orthographic projection can be a combination of viewpoint-independent and viewpoint-dependent edges. Henceforth, in this discussion, we will restrict our attention to viewpoint-dependent depth discontinuities.

We had previously promised a justification for the absence of curvature discontinuities at depth discontinuities which project to conic-sections. The argument runs as follows. We choose four depth discontinuities and perturb three of them keeping the fourth one fixed. This can always be done by choosing the projection of the surface normal at the fourth point to be the axis of perturbation. Now note that r_{xi} at the fourth will be well defined by the three other r_{xi} because the three depth discontinuities must be non-collinear. But, the fourth point could have been chosen at any depth discontinuity. It follows that the normal curvature in the direction of projection is well defined at every depth discontinuity which projects onto a conic-section. By the same arguments as those presented for the straight-line case, we deduce that the surface is locally C^2.

If the orientation of the plane of the conic-section in space were known, we could obtain expressions for x_i and y_i in terms of the angle ϕ_i formed by the tangent to this curve with a fixed axis. Then, we must be able to choose α, β and γ such that the right-hand-side of (17) is independent of ϕ_x. This choice, if possible, would determine r_i up to a multiplicative factor. Thus, the normal curvature of the surface along the depth discontinuity curve, in the direction of projection, has three degrees of freedom, two for the orientation of the plane of the conic-section in space and one for the multiplicative factor. Given the orientation of the plane of the space conic, the curvature of the surface along the depth discontinuity curve, in the direction of the tangent along the conic, can also be found. It may seem, at first, that because we only know the normal curvatures of the surface in two directions at each depth discontinuity, we are still one constraint short of being able to determine the principal curvatures and their orientations. But, the normal curvature in a third direction at any point may be determined by using the fact that the surface is C^2 and that the depth of a point along the third direction can be expressed as a function of

the normal curvature in the projection direction at an adjacent point. (Higher order terms are disregarded as before.) Thus, the first and second fundamental forms (see [Lipschutz, 1969]) of the surface are completely determined, up to three degrees of freedom, along the depth discontinuity curve.

The reader may wonder if there do exist continuous-surface-tangent surfaces which project onto ellipses, parabolas, and hyperbolas, under *general* viewpoint. There do. The simplest examples are members of the quadric family. We show this in the Appendix. In fact, if we arbitrarily fix the depth of the quadric from the projection plane, then the quadric which orthographically projects onto a given quadratic, is completely determined up to the three degrees of freedom specified above. This too is shown in the Appendix. The fact that quadrics project onto quadratics implies that that the local second order descriptions of surfaces which project onto ellipses are either ellipsoids or hyperboloids of one sheet; which project to parabolas are either elliptic paraboloids or hyperbolic paraboloids; which project to hyperbolas are either hyperboloids of one sheet or hyperboloids of two sheets. The classification of quadrics can be found in [Olmsted, 1947].

VI. Circles

Theorem 3 : *Circles in a line-drawing obtained under orthographic projection with a* general *viewpoint are projections of circles in space which are confined to lie in planes parallel to the image plane.*

Proof : We proceed in the same fashion as in the proof of the previous theorem except that the column with the $x_i y_i$ terms in **P** is now absent and so we need only consider five points instead of six. We finally arrive at the following requirement instead of equation (15).

$$[x_5^2 \;\; y_5^2 \;\; x_5 \;\; y_5 \;\; 1]\, \mathbf{S}^T (\mathbf{S}\mathbf{S}^T)^{-1} \mathbf{T}' - [2y_5 \;\; 1] z_5 + [p_1]\, \mathbf{U} = 0 \qquad (18)$$

where $\mathbf{S} = \begin{pmatrix} x_1^2 & y_1^2 & x_1 & y_1 & 1 \\ x_2^2 & y_2^2 & x_2 & y_2 & 1 \\ x_3^2 & y_3^2 & x_3 & y_3 & 1 \\ x_4^2 & y_4^2 & x_4 & y_4 & 1 \end{pmatrix}$

$$\mathbf{T}' = \begin{pmatrix} 2y_1 z_1 & z_1 \\ 2y_2 z_2 & z_2 \\ 2y_3 z_3 & z_3 \\ 2y_4 z_4 & z_4 \end{pmatrix}$$

and **U** *depends on* (x_i, y_i), $i = 1..4$.

Going through a similar line of reasoning as before, we may conclude that the depth coordinates z_i are of $(by_i + c)$ form. But, if our perturbation was about the y-axis instead of about the x-axis, we would instead have concluded that z_i are of $(ax_i + c)$ form. Because the expressions for z_i must be the same in both cases, it follows that the depth z_i of the points is constant. Finally, note that curves confined to planes perpendicular to the projection direction, orthographically project onto identical curves. This completes the proof.

Q.E.D.

Corollary : *Circles in a line-drawing obtained under orthographic projection with a* general *viewpoint are projections of continuous-surface-tangent depth discontinuities in space.*

As we did for general conics above, we now constrain the surface curvatures along the depth discontinuity curve, in this case a circle. We once again consider the expression (17) for r_i. If the radius of the circle is r, then without any loss of generality we can substitute, $x_i = x_o + r\cos\phi_i$, and $y_i = y_o + r\sin\phi_i$. This leads to the result that $r_i = mr$ where m is an arbitrary multiplicative factor. The conclusion that r_i is constant could also be drawn purely from symmetry considerations.

Now, we point out that the satisfaction of a constraint of the type (12) is not sufficient to ensure that the mapping under perturbation of viewpoint will remain a circle. We also require that the coefficients of the x^2 and y^2 terms in the quadratic remain equal. This requirement, after some algebra, can be shown to be unsatisfiable under the second order approximation (1). However, it can be

verified using the constraints derived above that the multiplicative factor m, in the expression for r_i, must be unity if the circular shape is to be maintained under perturbation of viewpoint. Thus, we are lead to the conclusion that every point on the space curve which projects to a circle is an umbilical point (see [Lipschutz, 1969]) with radius of curvature equal to the radius of the projected circle. Hence, the local second order description of the surface is a sphere with the same radius as that of the projected circle. Notice that the first and second fundamental forms of the surface are completely determined, without any degree of freedom, along the depth discontinuity curve.

VII. Four More Theorems

Before we state the theorems, we point out that the analyses and results presented throughout this paper are valid even if the straight-lines and conic-sections in the line-drawing are fragmented.

Theorem 4 : *Scene events which orthographically project, under* general *viewpoint, onto straight-lines or conic-sections in line-drawings must either be completely viewpoint-independent or completely viewpoint-dependent.*

Theorem 5 : *Continuous-surface-tangent depth discontinuities which orthographically project, under* general *viewpoint, onto straight-lines in line-drawings can be described locally by developable surfaces.*

Theorem 6 : *Continuous-surface-tangent depth discontinuities which orthographically project, under* general *viewpoint, onto conic-sections in line-drawings can be described locally by quadric surfaces. Specifically, continuous-surface-tangent depth discontinuities projecting onto ellipses by ellipsoids or hyperboloids of one sheet, onto parabolas by elliptic paraboloids or hyperbolic paraboloids, and onto hyperbolas by hyperboloids of one sheet or hyperboloids of two sheets. Furthermore, these quadric surfaces are completely determined, up to three degrees of freedom, by the conic-sections in the line-drawings.*

Theorem 7 : *Depth discontinuities which orthographically project, under* general *viewpoint, onto circles in line-drawings can be described locally by spheres whose radii are identical to the radii of the circles in the line-drawings.*

In Theorem 4, viewpoint-independent scene events refer to combinations of surface-tangent discontinuities, surface-reflectance discontinuities and illumination discontinuities, while viewpoint-dependent scene events refer solely to continuous-surface-tangent depth discontinuities. As mentioned previously, pairs of straight-lines are considered to be instances of straight-lines rather than instances of degenerate conic-sections.

Theorem 4 immediately follows from the discussion so far. Theorem 5 was proved in the section on straight-lines. Theorem 7 follows from Theorem 6 and the discussion in the section on circles. We now prove Theorem 6.

Proof of Theorem 6 : In section V, it was shown that continuous-surface-tangent depth discontinuities which orthographically project, under *general* viewpoint, onto conic-sections in line-drawings can be described up to second order properties by local quadric surfaces. We now seek to show that the quadric description is actually valid for properties of all orders, not just the second order. We use the following result from a paper by Maschke [Maschke, 1902] : *The developable surfaces and the surfaces of the second order are then the only ones for every point of which a surface of the second order exists having a contact of the third order.* It follows from this result that all we need to establish is third order contact between the projected surface and a quadric at every point local to the depth discontinuity curve on the surface.

Under, the *general* viewpoint assumption only isolated points on the depth discontinuity curves may be boundary points of a C^3 patch. Let us consider a depth discontinuity point in the interior of a C^3 patch. With an appropriate choice of the coordinate frame, the surface may be locally expressed by the the following Taylor series expansion about the origin. The coordinate axes are u, v, and w, and the surface is tangent to the u-v plane at the origin. L, M, and N are the coefficients of the second fundamental form of the surface (see

[Lipschutz, 1969]).

$$w = \frac{1}{2}(Lu^2 + 2Muv + Nv^2) + \frac{1}{6}(Pu^3 + 3Qu^2v + 3Ruv^2 + Sv^3) + o([u^2 + v^2]^{3/2}) \quad (19)$$

Let the projection direction lie in the u-v plane and make a counter-clockwise angle θ with the u-axis. Then , along the depth discontinuity curve, $\left[-\sin(\theta)\frac{\partial w}{\partial u} + \cos(\theta)\frac{\partial w}{\partial v}\right]$ must be zero.

$$-\sin(\theta)\left[(Lu + Mv) + \frac{1}{2}(Pu^2 + 2Quv + Rv^2) + ..\right] + \cos(\theta)\left[(Mu + Nv) + \frac{1}{2}(Qu^2 + 2Ruv + Sv^2) + ..\right] = 0 \quad (20)$$

This equation implicitly expresses v in terms of u. Now we use the fact that the depth discontinuity curve must be planar in space. Equivalently, the torsion of the space curve $\mathbf{p} = [u\ v\ w]$ must be zero, i.e., $\det\left([\mathbf{p}'\ \mathbf{p}''\ \mathbf{p}''']^T\right)$ must equal zero (see [Lipschutz, 1969]). If we parameterize the curve in terms of u, then we will get

$$v''w''' - v'''w'' = 0. \quad (21)$$

We require that this equation be satisfied at the origin. If we differentiate equations (20) and (19), we get the following equalities at the origin.

$$v' = \frac{L\sin(\theta) - M\cos(\theta)}{N\cos(\theta) - M\sin(\theta)}$$

$$v'' = \frac{[P + 2Qv' + Rv'v']\sin(\theta)}{N\cos(\theta) - M\sin(\theta)} - \frac{[Q + 2Rv' + Sv'v']\cos(\theta)}{N\cos(\theta) - M\sin(\theta)}$$

$$v''' = \frac{3[Q + Rv']\sin(\theta) - 3[R + Sv']\cos(\theta)}{N\cos(\theta) - M\sin(\theta)}\ v''$$

$$w'' = (L + Mv' + Nv'v')$$

$$w''' = 3(M + Nv')v'' + (P + 3Qv' + 3Rv'v' + Sv'v'v')$$

It follows from these equations that v'' or $(w''' - w''v'''/v'')$, both of which are linear combinations of P, Q, R, and S, must be zero. Equivalently, if we put $A = P + Qv'$, $B = Q + Rv'$, and $C = R + Sv'$, then at least one of the following two equations must be true.

$$(A + Bv')\sin(\theta) - (B + Cv')\cos(\theta) = 0 \quad (22)$$

$$3(M + Nv')\left[(A + Bv')\sin(\theta) - (B + Cv')\cos(\theta)\right] + (N\cos(\theta) - M\sin(\theta))\left[A + 2Bv' + Cv'v'\right] - 3w''\left[B\sin(\theta) - C\cos(\theta)\right] = 0 \quad (23)$$

Notice that both the equations are homogeneous linear equations. Assume that L, M, and N are known at the origin. Now, if perturbation of θ leads to only one independent equation, then P, Q, R, and S have three degrees of freedom, i.e., we can choose three of these coefficients arbitrarily. This can only happen if the depth discontinuity curve is locally fixed in space under perturbation of viewpoint. Then, v' must be constant and the surface must therefore be parabolic at the origin, i.e., $LN - M^2 = 0$. In fact, the surface must be locally developable. As projections of such depth discontinuities will have zero curvature in the line-drawing, we need not consider them here. Hence, the number of linearly independent equations must either be two or three. If it is two, then P, Q, R, and S have two degrees of freedom, and if it is three, then there is only one degree of freedom. It can easily be verified by differentiation that the third derivatives of a quadric surface with known L, M, and N, and constrained to be tangential to the u-v plane at the origin, have two degrees of freedom. Further, we also know that the constraints between P, Q, R, and S are satisfied by every quadric surface because, as shown in the Appendix, the depth discontinuity curve on a quadric is always planar. Hence we can always find a quadric surface which has third order contact with the projected surface at the origin.

We have established third order contact between a quadric surface and the projected surface at the depth discontinuity under consideration. By moving our depth discontinuity to any interior point of a C^3 surface patch, we can establish third order contact with a quadric surface at every such point. From Maschke's result it follows that the surface can be locally described by quadric patches, one for each C^3 patch along the depth discon-

tinuity curve. But we have previously established that a single quadric surface has second order contact along the whole depth discontinuity curve. Further, it was shown in the Appendix that this quadric is completely determined up to three degrees of freedom. The desired result immediately follows.

Q.E.D.

There is a result by Blaschke [Blaschke, 1923] that the only surfaces which yield planar depth discontinuity curves under orthographic projection for all viewpoints are the quadrics and the developable surfaces. His proof implicitly assumes the surface to be C^4. In our proof of Theorem 6, we have essentially generalized Blaschke's result to the following. All surfaces which yield planar depth discontinuity curves under orthographic projection for all viewpoints within an open set on the Gaussian sphere can be described locally by the quadrics and the developable surfaces. Our proof assumes a C^3 surface.

VII. Conclusion

Line-drawing interpretation refers to the attempt to infer 3-D shape information about the scene from its line-drawing. We view the problem to be one of surface constraint generation under the assumption of a *general* viewpoint. As mentioned previously, the *general* viewpoint assumption is essential to make any headway. Without this assumption it is not possible to make non-trivial deductions. In this work we attempted to discover constraints associated with straight-lines and conic-sections in line-drawings obtained under orthographic projection. Necessary and sufficient conditions were derived in each case.

Related previous work of interest includes that of Whitney [Whitney, 1955] and Koenderink [Koenderink, 1984]. It follows from Whitney's paper that there are only two types of *generic* singularities in the projection of a C^3 surface onto a plane. They are what he calls "folds" and "cusps." "Folds" are depth discontinuity curves and "cusps" are their terminations. Whitney's results are valid for both, perspective and orthographic projections. However, his *genericity* condition is more relaxed than our *general* viewpoint condition in that it demands stability under perturbation of both, the viewpoint and the projected surface. Koenderink has shown that a depth discontinuity is an elliptic, hyperbolic, or parabolic point on the viewed surface depending on whether its projected image in the line-drawing is convex, concave or an inflection, respectively. His work makes the same assumptions as Whitney's.

Finally, we note that we have made no effort whatsoever to suggest schemes for the detection of straight-lines and conic-sections in line-drawings. Neither have we analyzed the robustness of the conclusions which may be drawn from such instances, i.e., how sensitive are the surface predictions to errors in the detection and parameterization of curves in line-drawings. Practical considerations demand that these issues be addressed.

Appendix

We first show that the orthographic projection of a quadric surface is always a quadratic curve. To see this, it is sufficient to verify that the depth discontinuity curves on quadrics are always planar. Let us assume the projection geometry of Fig. 3. Any quadric can always be considered the zero potential surface of some second order potential field,

$$C(x, y, z) = ax^2 + by^2 + cz^2 + dxy + exz + fyz + gx + hy + iz + j .$$

Then the normal to the quadric, C(x,y,z) = 0, at any point is in the direction of the potential gradient at that point. Now observe that a point on the surface is a depth discontinuity if and only if the surface normal at that point is orthogonal to the projection direction, i.e., the z-axis here. Hence, the depth discontinuity curve must satisfy the following equation.

$$\frac{\partial C(x, y, z)}{\partial z} = 2cz + ex + fy + i = 0$$

This is the equation of a plane. The orthographic projection of our quadric, C(x, y, z) = 0, can be simply obtained by substituting for z from the equation of the plane. The resulting curve is

immediately seen to be a quadratic.

Now we show that if we arbitrarily fix the depth of the quadric surface from the projection plane then the surface is completely determined, up to three degrees of freedom, by the projected quadratic. Let us assume that the equation of the plane containing the space curve is $z + \alpha x + \beta y = 0$. This equation has two degrees of freedom. Constraining this plane to pass through the origin fixes the depth of the quadric surface. Equating the coefficients of this plane with those of the plane derived above, we get $e = 2\alpha c$, $f = 2\beta c$, $i = 0$. Now, if we substitute for z into $C(x, y, z) = 0$, we will get

$$(a - \alpha^2 c)x^2 + (b - \beta^2 c)y^2 + (d - 2\alpha\beta c)xy + gx + hy + j = 0.$$

But as we know the equation of the projected quadratic, we can determine all the coefficients of the quadric $C(x, y, z) = 0$ in terms of one unknown c. This unknown determines the scaling factor of the radii of normal curvature along the depth discontinuity curve on the surface (see Section V).

Acknowledgement

The author is deeply indebted to Raz Stowe for substantial mathematical assistance. He also wishes to thank Tom Binford for his support. This work was supported in part by the Defense Advanced Research Projects Agency under contract N00039-84-C-0211.

References

Blaschke, W., *Vorlesungen über Differentialgeometrie II*, Springer, Berlin, 1923.

Coddington, E.A., *An Introduction to Ordinary Differential Equations*, Prentice-Hall, New Jersey, 1961.

Coxeter, H.S.M., *Introduction to Geometry*, John Wiley & Sons, New York, 1961.

Golubitsky, M., and Guillemin, V., *Stable Mappings and Their Singularities*, Springer-Verlag, New York, 1973.

Guillemin, V., and Pollack, A., *Differential Topology*, Prentice-Hall, New Jersey, 1974.

Hilbert, D., and Cohn-Vossen, S., *Geometry and the Imagination*, Chelsea, New York, 1952.

Koenderink, J.J., "What does the occluding contour tell us about solid shape?," *Perception*, Vol. 13, 1984, 321-330.

Lipschutz, M.M., *Differential Geometry*, McGraw-Hill, New York, 1969.

Maschke, H., "On Superosculating Quadric Surfaces," *Transactions of the American Mathematical Society*, Vol. 3, 1902, 482-484.

Olmsted, J.M.H., *Solid Analytic Geometry*, Appleton-Century-Crofts, New York, 1947.

Strang, G., *Linear Algebra and Its Applications*, Academic Press, New York, 1976.

Whitney, H., "On Singularities of Mappings of Euclidean Spaces. I. Mappings of the Plane into the Plane," *Annals of Mathematics*, Vol. 62, Nov. 1955, 374-410.

Since submitting the paper, the author has discovered that the proof of theorem 6 is incorrect. Correct proof will be presented in a subsequent version.

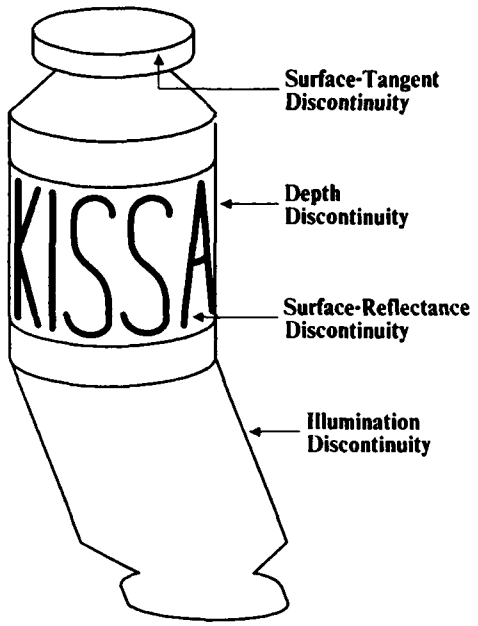


Fig. 1. Line-Drawing with Various Types of Edges

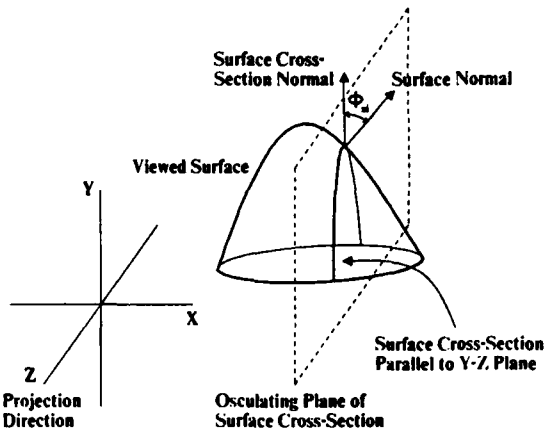


Fig. 3. The Projection Geometry

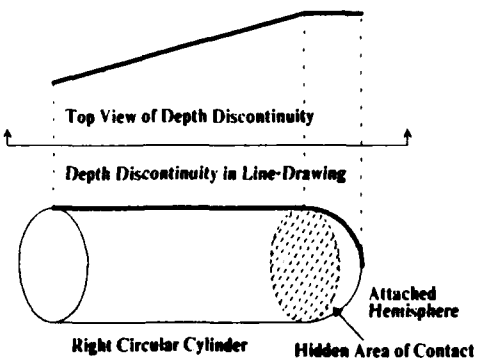


Fig. 2. A Depth Discontinuity with a Continuous Tangent in a Line-Drawing need not have a Continuous Tangent in Space

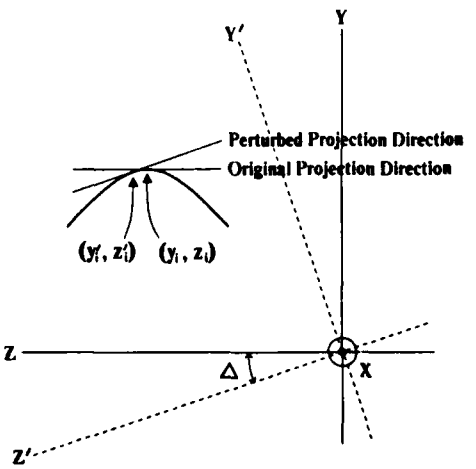


Fig. 4. Change in Depth Discontinuity under Perturbation of Viewpoint

Line-Drawing Interpretation : Bilateral Symmetry

Vic Nalwa

A.I. Lab., Stanford University, CA 94305

Abstract

Symmetry manifests itself in numerous forms throughout nature and the man-made world. It is frequently encountered in line-drawings corresponding to edges in images. We seek to discover constraints on the imaged surfaces from instances of bilateral (reflective) symmetry in line-drawings obtained from orthographically projected images. A general *viewpoint is assumed, i.e., it is assumed that the mapping of the viewed surface onto the line-drawing is stable under perturbation of the viewpoint within some open set on the Gaussian sphere. It is shown that the only planar figures which exhibit bilateral symmetry under orthographic projection with a* general *viewpoint are ellipses and straight-line segments.*

We show that the orthographic projection of a surface of revolution exhibits bilateral symmetry about the projection of the axis of revolution, irrespective of the viewing direction. Further, it is shown that whenever the symmetry axis is the projection of a line in space which is invariant under perturbation of the viewpoint, the bilaterally symmetric line-drawing is the orthographic projection of a local surface of revolution. The axis of revolution is the back-projection of the symmetry axis. Various line-drawing configurations for which the back-projection is invariant are detailed. It is conjectured that isolated ellipses and isolated straight-line segments are the only bilaterally symmetric line-drawings whose symmetry axes are not projections of lines in space which are invariant under perturbation of the viewpoint. Throughout, only surface regions local to the scene-events corresponding to the lines are constrained.

I. Introduction

Line-drawings are representations of edges in an image. An edge in an image may be caused by various events in the scene. The scene-event may be a surface-tangent discontinuity, a depth discontinuity, a surface-reflectance discontinuity or an illumination discontinuity (shadow). Fig. 1 illustrates the various cases. Notice that a depth discontinuity may also simultaneously be a surface-tangent discontinuity. We distinguish this from a continuous-surface-tangent depth discontinuity. While surface-tangent discontinuities, illumination discontinuities, and surface-reflectance discontinuities constitute viewpoint-independent scene-edges, continuous-surface-tangent depth discontinuities are viewpoint-dependent.

An object exhibits symmetry whenever it can be divided into two or more parts which can be permuted by the application of certain isometries which leave the original object unchanged. Symmetry manifests itself in numerous forms throughout nature and the man-made world. Some examples are shown in Fig. 2. The two symmetry operations for planar figures are reflection and rotation. The letter A is an instance of a figure with reflective symmetry and the letter Z exhibits rotational symmetry. The letter H admits both symmetry operations. Three dimensional space admits several symmetry operations in addition to reflection and rotation, e.g., inversion symmetry is exhibited by a pair of bicycle cranks. In this presentation we are concerned with the three dimensional implications of bilateral (reflective) symmetry in a line-drawing.

Line-drawings, like images, are many-to-one mappings from a 3-D domain to a 2-D range.

Although the depth information is lost under projection, we can nevertheless impose some constraints on the 3-D scene by investigating the 2-D configuration of the corresponding line-drawing. We restrict our attention to orthographic projection here. Orthographic projection is a reasonable approximation to perspective projection whenever the angle subtended by the object at the viewpoint is small. Orthographically projected line-drawings of man-made scenes often exhibit instances of bilateral symmetry. We investigate constraints imposed on the scene by such instances under the assumption of *general* viewpoint, i.e., under the assumption that the mapping of the viewed surface onto the line-drawing is stable under perturbation of the viewpoint within some open set on the Gaussian sphere[1]. The stability we demand includes the requirement that the bilateral symmetry exhibited by the line-drawing be preserved under perturbation of viewpoint. Other properties of the mapping with respect to which we require stability will be mentioned as we proceed. As long as the total number of such requirements is finite, the set of viewing directions on the Gaussian sphere where the *general* viewpoint assumption is invalid is closed and has measure zero[2]. It must be emphasized here that we are not seeking *ad hoc* hueristics. We seek to discover rigorous constraints under specific reasonable assumptions. It should be noted that, given the loss of information about the third dimension under projection, one cannot hope to make any substantial deductions about the imaged surfaces without the *general* viewpoint assumption.

Related previous work of interest includes that of Kanade [Kanade, 1981] and Marr [Marr, 1977]. Kanade presents a heuristic which infers symmetry in the scene whenever skewed symmetry is discovered in an orthographically projected line-drawing of a world containing only planar surfaces. Skewed symmetry is meant to indicate planar shapes in which symmetry is exhibited along lines at a fixed angle to the axis of symmetry, e.g., see Fig. 3. Of course, if this fixed angle is a right angle then we have bilateral symmetry. Although planar bilateral symmetries do map onto skewed symmetries, skewed symmetries also map onto skewed symmetries. Hence, Kanade's inference requires skewed symmetries to be excluded from his world of planar surfaces. Marr [Marr, 1977] uses "qualitative symmetry" to deduce generalized cones, i.e., solids whose cross-sections perpendicular to an axis are all scaled versions of one another. His concept of qualitative symmetry, although more relaxed than our concept of bilateral symmetry, does not lead to the conclusions he wishes to draw under the *general* viewpoint paradigm presented here. Our work is more in the spirit of Koenderink and van Doorn who have shown in two elegant articles that the Gaussian curvature of a body can be deduced from the curvature of the occluding contour [Koenderink, 1984] and that terminations of occluding contours are always concave [Koenderink and van Doorn, 1982].

In Section II we show that the line-drawing of an orthographically projected surface of revolution exhibits bilateral symmetry about the projection of the axis of revolution, irrespective of the viewing direction. Section III investigates whether local surfaces of revolution are necessary for bilateral symmetry in a line-drawing obtained under *general* orthographic projection. It is shown that whenever the symmetry axis is the projection of a line in space which is invariant under perturbation of the viewpoint, the bilaterally symmetric line-drawing is

[1] The Gaussian sphere is a device to represent orientation in space. Each point on a unit sphere, called the Gaussian sphere (see [Hilbert and Cohn-Vossen, 1952]), corresponds to the unit vector from the center of the sphere to that point.

[2] An arbitrary set in any manifold has measure zero if its preimage, for every local parametrization, can be covered by a countable number of rectangular solids with an arbitrarily small total volume in Euclidean space. For our purposes it suffices to note that the Gaussian sphere is a two dimensional manifold and consequently all sets of points and lines on it have measure zero. The null set of course has measure zero. On the other hand, no open set on the Gaussian sphere has measure zero.

It can be shown that the union of a countable number of sets of measure zero is also of measure zero (see [Guillemin and Pollack, 1974]). Therefore, the union of a finite number of closed sets of measure zero is closed and of measure zero. The complement of such a union is open and of full measure.

the orthographic projection of a local surface of revolution. The axis of revolution is the back-projection of the symmetry axis. Various line-drawing configurations for which the back-projection is invariant are detailed. It is conjectured that isolated ellipses and isolated straight-line segments are the only bilaterally symmetric line-drawings whose symmetry axes are not projections of lines in space which are invariant under perturbation of the viewpoint. Throughout, only surface regions local to the scene-events corresponding to the lines are constrained. We conclude with Section IV.

The Appendix contains a proof for the proposition that segments of straight-lines and conic-sections are the only planar curves whose orthographic projections exhibit local bilateral symmetry under *general* viewpoint. It immediately follows that ellipses and straight-line segments are the only finite planar curves whose orthographic projections exhibit (global) bilateral symmetry under *general* viewpoint.

II. Surface of Revolution : A Sufficient Condition.

It is easy to see that an object displaying bilateral symmetry in space need not exhibit bilateral symmetry under orthographic projection (e.g., Fig. 2-c). Fig. 3 shows that bilaterally symmetric planar curves, in general, project to what Kanade [Kanade, 1981] calls skewed symmetric contours. As shown in the Appendix, ellipses and straight-line segments are the only exceptions to this phenomenon in that they project to bilaterally symmetric curves.

We now seek to argue that the orthographic projection of any surface of revolution, i.e., a surface generated by revolving a curve (called the generator) about an axis, will always exhibit bilateral symmetry about the projection of the axis of revolution. Consider such a surface to have its axis aligned with an arbitrary vector in space. Fig. 4 shows one such vector. Its orientation is completely determined by the specification of its azimuth angle, i.e., the counter-clockwise angle made by its projection onto the x-y plane with the x axis, and its polar angle, i.e., the angle it makes with the z axis. It is easily seen that the symmetry properties of the projection onto the x-y plane are not affected by the azimuth angle. Hence, without loss of generality we can always consider it to be zero. Now, if we note that the z-x plane divides the surface of revolution into two mirror images, we can immediately deduce that the orthographic projection will exhibit bilateral symmetry about the x axis. Fig. 5 shows an example of the orthographic projection of a surface of revolution. The mirror line is the projection of the axis of revolution.

The above analysis assumes that the surface of revolution is not obscured by some other object and that it has no shadow across it. Also, surface marks, if any, must be uniform over sub-surfaces which themselves are surfaces of revolution with the same axis as the complete surface.

III. Surface of Revolution : A Necessary Condition?

Is a surface of revolution a necessary condition for a bilaterally symmetric orthographic projection under *general* viewpoint? Obviously not. We can imagine arbitrary surfaces, without any scene-edges, corresponding to regions of a symmetric line-drawing which are "distant" from the lines. These surfaces can always be chosen such that small arbitrary perturbations of the viewpoint do not violate bilateral symmetry in the projection. As no information about these regions, other than the absence of edges, is available in the line-drawing, it is not possible to deduce stronger constraints for such regions without making assumptions regarding the nature of the imaged world. Therefore, we restrict our attention to the regions of surfaces local to the edges.

We now argue that whenever the symmetry axis is the projection of a line in space which is invariant under perturbation of the viewpoint, the bilaterally symmetric line-drawing is the orthographic projection of a local surface of revolution. The axis of revolution is the back-projection of the symmetry axis. Let us assume that the back-projection of the symmetry axis is oriented along the vector shown in Fig. 4 and that once again the

orthographic projection is along the z-axis and that the azimuth angle is zero. Now we point out that the *general* viewpoint assumption can be translated to the requirement that bilateral symmetry be preserved under perturbation of the imaged surface position. It is sufficient to consider combinations of two such perturbations : one, rotation of the surface about the y-axis – we call this *tilting*, and two, rotation of the surface about the back-projection of the symmetry axis – we call this simply rotation.

First, we argue that if the orthographic projection of the viewed surface is to remain bilaterally symmetric under tilting, then the surface must exhibit local bilateral symmetry in space about the x-z plane. To see this, consider the intersections of the viewed surface with any two planes parallel to the x-z plane and passing through a pair of bilaterally symmetric points in the line-drawing. Every such pair of intersection curves must be locally symmetric about the x-z plane. If the planes do not intersect the viewed surface, but are rather tangential to it, then the two tangent points or straight-line segments must also be bilaterally symmetric. Hence, the viewed surface must exhibit local bilateral symmetry. Further, because we could always conceive of a surface perturbation which is a rotation followed by tilting, we require that the surface maintain its local bilateral symmetry about the x-z plane when it is subject to rotation. But this requirement can only be satisfied by local surfaces of revolution about the back-projection of the symmetry axis. To see this, consider the following argument. For a surface to be bilaterally symmetric about a plane, its intersection with any plane orthogonal to the symmetry plane must also exhibit bilateral symmetry. In particular, consider the planes which have their normals oriented along the back-projection of the symmetry axis. The requirement that bilateral symmetry of the surface be maintained under rotation is equivalent to the requirement that all the described intersections maintain their symmetry in space. As the only such planar curves are circles, the argument is complete.

The result just derived is of practical significance only if we have mechanisms to deduce invariance of the back-projections of the symmetry axis from the line-drawing itself. As we will use results derived in [Nalwa, 1987] for this purpose, we need to to make the same assumptions about the viewed surfaces as are made there. In particular, we assume that the projected surfaces are piecewise C^3, i.e., the surfaces comprise finitely many C^3 patches, each bounded by a finite piecewise C^3 curve. This restriction does not significantly constrain the domain as long as we do not require a priori knowledge of the C^3 surface-patch boundaries.

The piecewise C^3 assumption enables us to draw the following conclusions (see [Nalwa, 1987]). A straight-line in a line-drawing is either the projection of a straight viewpoint-independent edge or of a straight viewpoint-dependent edge. If the edge is viewpoint-dependent then it can be locally described by a quadric cylinder or a quadric cone. An elliptical segment in a line-drawing is either the projection of an elliptical viewpoint-independent edge or of an elliptical viewpoint-dependent edge. If the edge is viewpoint-dependent then it can be locally described by an ellipsoid or a hyperboloid of one sheet. These conclusions hold even if the straight-line and elliptical segments in the line-drawing are fragmented.

Now we detail various line-drawing configurations for which the back-projection of the symmetry axis is invariant under perturbation of the viewpoint. Case I. Consider a bilaterally symmetric line-drawing which has two parallel straight-line segments, one on either side of the symmetry axis. For stable bilateral symmetry the two straight scene-edges must be viewpoint-dependent and further must locally lie on a single right-circular cylinder. The back-projection of the symmetry axis is the axis of this cylinder. Case II. Consider a bilaterally symmetric line-drawing which has two non-parallel straight-line segments, one on either side of the symmetry axis. For stable bilateral symmetry the two straight scene-edges must be viewpoint-dependent and further must

locally lie on a single right-circular cone. The back-projection of the symmetry axis is the axis of this cone. Case III. Consider a bilaterally symmetric line-drawing which has an elliptical segment (may be fragmented) bisected by the symmetry axis. It is easy to see that the back-projection of the centroid of the (completed) ellipse must continue to lie on the back-projection of the symmetry axis under perturbation of the viewpoint. Similarly, a tangent-discontinuity on the symmetry axis in a line-drawing must be the projection of a conical tip in space which continues to lie on the back-projection of the symmetry axis under perturbation of the viewpoint. As any two points in space determine a line, two independent events from among ellipses bisected by the symmetry axis and tangent-discontinuities on the symmetry axis fix the back-projection of the symmetry axis. Further investigation would almost certainly lead to more relaxed conditions for the invariance of the back-projection of the symmetry axis. Fig. 6 provides several examples of bilaterally symmetric line-drawings for which we can presently deduce local surfaces of revolution.

It is our conjecture that isolated ellipses and isolated straight-line segments are the only bilaterally symmetric line-drawings whose symmetry axes are not projections of lines in space which are invariant under perturbation of the viewpoint. What we do know is that ellipses and straight-line segments are the only planar figures which exhibit bilateral symmetry under orthographic projection with a *general* viewpoint (see Appendix). Ellipses project onto ellipses and straight-line segments onto straight-line segments. It follows that all viewpoint-independent scene-edges which are isolated ellipses or isolated straight-line segments exhibit bilateral symmetry under projection. These, of course, are not local surfaces of revolution. The viewpoint-dependent scene-edges which project onto isolated ellipses are either local ellipsoids or local hyperboloids of one sheet (see [Nalwa, 1987]). These, in general, are also not local surfaces of revolution. From the result in [Koenderink and van Doorn, 1982], that terminations of occluding contours are always concave, it follows that no viewpoint-dependent scene-edge may project onto an isolated straight-line segment.

IV. Conclusion

Line-drawing interpretation refers to the attempt to infer 3-D shape information about the scene from its line-drawing. This line of work has a long tradition in both computer vision (see [Malik, 1985]) and psychology (e.g., [Attneave and Frost, 1969]). Most of the work in computer vision has concentrated on exhaustively listing the junction possibilities in a polyhedral world, although recently there have been attempts to handle simple curved objects too (e.g., [Malik, 1985]). Relatively few attempts have been made at exploiting the structure of curves in line-drawings (e.g., [Lowe and Binford, 1984]).

We view the line-drawing interpretation problem to be one of constraint-specification, i.e., generation of all possible constraints which can be deduced from a line-drawing under the *general* viewpoint assumption. This, in our view, should be followed by an attempt to seek 3-D spatial configurations compatible with the specified constraints. In order to limit the possibilities it may be necessary to introduce some restrictions on the world at this stage or to introduce the concept of "simple" and then try to find the "simplest" consistent interpretation.

In this work we have attempted to discover the constraints associated with bilateral symmetry in a line-drawing. It was shown that the orthographic projection of a surface of revolution exhibits bilateral symmetry about the projection of the axis of revolution, irrespective of the viewing direction. Further, it was shown that whenever the symmetry axis is the projection of a line in space which is invariant under perturbation of the viewpoint, the bilaterally symmetric line-drawing is the orthographic projection of a local surface of revolution. The axis of revolution is the back-projection of the symmetry axis. Various line-drawing configurations for which the back-projection is invariant were detailed. It was conjectured that isolated ellipses and isolated straight-line segments are the only bilaterally symmetric line-

drawings whose symmetry axes are not projections of lines in space which are invariant under perturbation of the viewpoint. Throughout, only surface regions local to the scene-events corresponding to the lines were constrained.

It might interest the reader to learn that this work was prompted by the observation in [Nalwa, 1987] that surfaces which orthographically project, under *general* viewpoint, onto circles in line-drawings can be described locally by spheres.

Finally, we note that we have made no effort whatsoever to suggest schemes for the detection in line-drawings of bilateral symmetry, in general, and ellipses and straight-lines, in particular. Neither have we analyzed the robustness of the conclusions which may be drawn from bilateral symmetry in line-drawings, i.e., whether approximate symmetry translates to approximate local surfaces of revolution. Practical considerations demand that these issues be addressed.

Appendix : Bilateral Symmetry in Orthographic Projections of Planar Curves

Theorem : *Segments of straight-lines and conic-sections, i.e., ellipses, parabolas, and hyperbolas, are the only planar curves whose orthographic projections exhibit local bilateral (reflective) symmetry under* general *viewpoint.*

Corollary 1 : *Curves whose orthographic projections exhibit local bilateral symmetry under* general *viewpoint are planar if and only if their projections are conic-sections or straight-lines.*

Corollary 2 : *Ellipses and straight-line segments are the only finite planar curves whose orthographic projections exhibit bilateral symmetry under* general *viewpoint.*

It is assumed throughout this appendix that the curves under consideration are piecewise analytic, i.e., the curves are comprised of finitely many analytic segments. This restriction does not in any significant way constrain the domain as long as we do not require a priori knowledge of the analytic segment end-points.

Note that the theorem refers to local bilateral symmetry, i.e. reflective symmetry exhibited on some open interval of the curve-segment. Corollary 2, on the other hand, refers to symmetry exhibited by the complete curve. It is assumed in the statement of the theorem that a pair of straight lines meeting at a point is a special case of a hyperbolic branch. It is of interest to note that the orthographic projections of conic-sections and straight-lines exhibit global bilateral symmetry under every viewpoint, not just some *general* viewpoint. Corollary 1 follows from the fact that the orthographic projection of a curve under *general* viewpoint is a conic-section or straight-line if and only if the original curve is a conic-section or straight-line.

It is our conjecture that the planarity restriction from Corollary 2 may be removed, i.e., we should be able to assert that ellipses and straight-line segments are the only finite space curves whose orthographic projections exhibit bilateral symmetry under *general* viewpoint. However, the planarity restriction cannot be removed from the Theorem. The circular helix is an example of a non-planar space curve whose segments exhibit local bilateral symmetry under orthographic projection with a *general* viewpoint.

Fig. 7 shows a curve-segment whose orthographic projection under *general* viewpoint is assumed to exhibit local bilateral symmetry. It is further assumed that for some viewpoint, the back-projection of the mirror axis is M. It intersects the curve at (x,y). It is easy to see that the tangent, T, at (x,y) will project to the tangent at the projection of (x,y). As the tangent at the projection of (x,y) is perpendicular to the mirror axis, any line parallel to this tangent will intersect (if at all) the projected curve at equal distances from the mirror line. Because, distances in any one direction on the object plane are uniformly distorted under projection, it follows that any line parallel to T will intersect (if at all) the original curve at two points, say (x_1,y_1) and (x_2,y_2), whose midpoint lies on M. This is a necessary and sufficient condition for the projected curve to exhibit local bilateral symmetry about the projection of M for some viewpoint. It has been assumed here that the

tangent, T, at (x,y) is well defined, i.e. the curve is C^1 at (x,y). If this is not true, then the above condition can be relaxed so that we require that every line parallel to some arbitrary line, T', intersect (if at all) the original curve at two points such that their midpoint lies on M.

Lemma 1 : *With the exception of a pair of straight-lines meeting at a point, planar curves whose orthographic projections exhibit local bilateral symmetry under* general *viewpoint are analytic and without inflections.*

Proof : Non-analytic and inflection points on any curve have a one-to-one correspondence with non-analytic and inflection points on the projected curve as long as the viewing direction does not lie in the plane of the original curve. Such viewing directions are excluded by the *general* viewpoint assumption. For the projected figure to exhibit bilateral symmetry we must be able to pair up non-analytic points and inflection points such that the pairs comprise of mirror images under projection. Every inflection must belong to one such pair. Non-analytic points, however, may also lie on the mirror axis itself. Bilateral symmetry requires that the mirror axis bisect the straight-lines joining the projections of the paired non-analytic (inflection) points. It follows that that the back-projection of the mirror axis must bisect the straight-lines joining the paired non-analytic (inflection) points in the object plane.

It is easy to see that in the presence of non-analytic (inflection) points, the back-projection of the mirror axis, if any, would be fixed in all cases except when there is either a single instance of paired non-analytic (inflection) points or an isolated non-analytic point on the mirror axis. With a fixed mirror axis, the curve cannot exhibit bilateral symmetry under *general* viewpoint because the angles formed at the intersection of the curve with the mirror axis cannot remain symmetric under perturbation of any viewpoint. Fig. 8 illustrates how a bilaterally symmetric curve with inflections and discontinuities undergoes distortion and loss of symmetry under orthographic projection.

In case the back-projection of the mirror axis is constrained to pass through the center of a line joining a pair of non-analytic (inflection) points, symmetry cannot be maintained under *general* viewpoint because of the following argument. The requirement that the line joining the pair of non-analytic (inflection) points be perpendicular to the mirror axis in the projected plane leads to the constraint that the back-projection of the mirror axis bisect all straight-line segments parallel to the line joining the pair of non-analytic (inflection) points and bounded by the original curve. But, for any given curve, this constraint fixes the back-projection of the mirror axis, if any. Hence, this configuration cannot exhibit symmetry under *general* viewpoint either.

In case of an isolated non-analytic point on the mirror axis we first note that the non-analytic point must be a tangent discontinuity. If this is not so, the necessary and sufficient condition described above cannot be satisfied under perturbation of the viewpoint. If the isolated discontinuity is a tangent discontinuity the only admissible local curve which would exhibit bilateral symmetry under *general* viewpoint is a pair of straight-lines. It follows that with the exception of a pair of straight-lines meeting at a point, the curve must be analytic and without inflections.

Q.E.D.

Lemma 2 : *With the exception of straight-lines, for planar curves whose orthographic projections exhibit local bilateral symmetry under* general *viewpoint, the back-projections of the axes of local symmetry under different viewpoints intersect at a single point.* (This point of intersection may be at infinity, i.e. all the back-projections may be parallel.)

Proof : If the curve is a pair of straight-lines meeting at a point, then this lemma is immediately seen to be true with all the back-projections of the mirror axes passing through the meeting point. If this is not so, then it follows from the previous lemma that the curve is analytic and without inflections. Consider Fig. 7, once again. As argued above, a necessary and sufficient condition for the projected curve to exhibit bilateral symmetry about the pro-

jection of M for some viewpoint is that any line parallel to the tangent, T, at (x,y) intersect (if at all) the original curve at two points, (x_1,y_1) and (x_2,y_2), whose midpoint lies on M. Now note that under perturbation of viewpoint within an open set on the Gaussian sphere, the back-projection of the mirror axis, in general, cannot continue to pass through (x,y) because bilateral symmetry requires that the projections of M and T intersect at right angles over and above the condition just specified. The only exception to this observation is the degenerate case of a straight-line for which every line through every point is the back-projection of an axis of symmetry for some viewpoint. Disregarding straight-lines, analyticity about (x,y) implies that a perturbation of the viewpoint will perturb the intersection, (x,y). We may assume, without any loss of generality, that the back-projection of the mirror axis under some different viewpoint passes through (x_1,y_1). Call this back-projection M_1 and let it intersect M at the origin O. If M_1 is parallel to M, then O will be at infinity. By the same arguments as those just presented, it is seen that any line parallel to the tangent, T_1, at (x_1,y_1) will intersect (if at all) the original curve at two points, say p_1 and q_1, such that their midpoint, r_1, lies on M_1. Now notice that, given the segment of the curve above M, the segment below M is completely determined. We may imagine it to be generated in the following way. First, project the upper segment orthographically in the direction for which the projection of M is the mirror axis. Then, reflect this projection about the mirror axis and back-project it onto the object plane. Let us now consider that this procedure is applied not only to the upper segment of the curve, but also to T_1, M_1, p_1, q_1 and r_1. Let the corresponding lines and points below M be subscripted by 2's. Then it is not hard to see that M_2 will pass through (x_2,y_2) and O. Also, T_2 will be tangent to the curve at (x_2,y_2). Further, p_2q_2 will be parallel to T_2 and r_2, the midpoint of p_2q_2, will lie on M_2. Consequently, M_2 is the back-projection of the mirror axis through (x_2,y_2). So, we have shown that three back-projections of mirror axes, M, M_1 and M_2, intersect at one point, O. As this argument holds for all choices of points and lines in the region above M, we can choose (x_1,y_1) arbitrarily close to (x,y). Then, we can proceed to apply the same argument to the point which is the mirror image of (x,y) about M_1 under orthographic projection, i.e. we make M_1 play the role previously played by M and conclude that the back-projection of the mirror axis through the new point will also pass through O. Continuation of this argument leads to the desired result.

Q.E.D.

Proof of Theorem : Consider Fig. 7, once again. We consider only curves other than straight-lines and a pair of straight-lines meeting at a point, both of which exhibit local bilateral symmetry under orthographic projection with a *general* viewpoint. Lemma 1 now allows us to restrict our attention to analytic curves without inflections and Lemma 2 tells us that the back-projections of the mirror axes either all intersect at a point in the finite plane or that they are all parallel. If the fixed point is not at infinity, we may assume without any loss of generality that it lies at the origin. As argued before, a necessary and sufficient condition for M to be the back-projection of a *mirror* axis is that any line parallel to T, the tangent at (x,y), intersect (if at all) the curve at two points, (x_1,y_1) and (x_2,y_2), whose midpoint lies on M. We express this observation in the following two equations. Equation (1) is obtained by equating the tangent at (x,y) to the slope of the line joining (x_1,y_1) and (x_2,y_2). Equation (2) is obtained by equating the slope of the back-projection of the mirror axis to that of the line joining (x,y) with the midpoint of (x_1,y_1) and (x_2,y_2). We call this slope s(x,y). It is a constant, k, when the back-projections of the mirror axes are parallel and y/x when they intersect at the origin.

$$\frac{dy}{dx} = \frac{y_1 - y_2}{x_1 - x_2} \tag{1}$$

$$s(x,y) = \frac{(y_1 + y_2)/2 - y}{(x_1 + x_2)/2 - x} \tag{2}$$

If we let $\Delta_1 = x_1 - x$ and $\Delta_2 = x_2 - x$, then we can rewrite the two equations as follows.

$$y_1 - y_2 = (\Delta_1 - \Delta_2)\frac{dy}{dx} \tag{3}$$

$$y_1 + y_2 = (\Delta_1 + \Delta_2)\, s(x,y) + 2y \qquad (4)$$

Adding and subtracting (3) and (4) we get,

$$2y_1 = 2y + (\Delta_1+\Delta_2)\, s(x,y) + (\Delta_1-\Delta_2)\frac{dy}{dx} \qquad (5)$$

$$2y_2 = 2y + (\Delta_1+\Delta_2)\, s(x,y) - (\Delta_1-\Delta_2)\frac{dy}{dx} \qquad (6)$$

We now write out the Taylor series expansion for y_1 and y_2. This, of course, is under the assumption that y is a function of x in the neighborhood of (x,y). This assumption is reasonable because were it not the case, we could always rotate the axes without any loss of generality.

$$y_1 = y + \Delta_1\frac{dy}{dx} + \frac{\Delta_1^2}{2}\frac{d^2y}{dx^2} + \frac{\Delta_1^3}{6}\frac{d^3y}{dx^3} + \textit{higher order terms} \qquad (7)$$

$$y_2 = y + \Delta_2\frac{dy}{dx} + \frac{\Delta_2^2}{2}\frac{d^2y}{dx^2} + \frac{\Delta_2^3}{6}\frac{d^3y}{dx^3} + \textit{higher order terms} \qquad (8)$$

We equate the expressions for y_1 in equations (5) and (7) to get equation (9) and the expressions for y_2 in equations (6) and (8) to get equation (10).

$$\Delta_2\left(s(x,y) - \frac{dy}{dx}\right) = \Delta_1\left(\frac{dy}{dx} - s(x,y)\right) + 2\left(\frac{\Delta_1^2}{2}\frac{d^2y}{dx^2} + \frac{\Delta_1^3}{6}\frac{d^3y}{dx^3} + \ldots\ldots\right) \qquad (9)$$

$$\Delta_1\left(s(x,y) - \frac{dy}{dx}\right) = \Delta_2\left(\frac{dy}{dx} - s(x,y)\right) + 2\left(\frac{\Delta_2^2}{2}\frac{d^2y}{dx^2} + \frac{\Delta_2^3}{6}\frac{d^3y}{dx^3} + \ldots\ldots\right) \qquad (10)$$

We now substitute for Δ_2 in terms of Δ_1 from equation (9) into equation (10) and equate the coefficients of the Δ_1^3 terms on the two sides of the resulting equation (the coefficients of the Δ_1 and Δ_1^2 terms result in identities). Simplification leads to the following third order non-linear ordinary differential equation.

$$\frac{d^3y}{dx^3}\left(\frac{dy}{dx} - s(x,y)\right) - 3\left(\frac{d^2y}{dx^2}\right)^2 = 0 \qquad (11)$$

The satisfaction of this third order ordinary differential equation in the neighborhood of the intersection of the back-projection of the mirror line and the original curve is a necessary condition for the projected curve to exhibit bilateral symmetry under perturbation of the viewpoint, i.e. under the *general* viewpoint assumption.

We now observe, that if the position, tangent and curvature of (x, y) completely determine any curve which also satisfies the differential equation, then this curve must be the unique solution to the third order differential equation under the given conditions. This is a result from the theory of ordinary differential equations[3]. Let us assume that the position, tangent and curvature of (x,y) are known. The curvature, κ, may be positive or negative. It may not be zero because straight-lines have been excluded from this discussion and inflections are disallowed by Lemma 1. We examine the various solutions to the differential equation for both, $s(x,y) = y/x$ and $s(x,y) = k$. First, we make two observations about restrictions on the position and tangent of (x,y). One, (x,y) cannot lie at the fixed point. It is not hard to see that this would make equations (1) and (2) unsatisfiable under perturbation of the viewpoint. Two, the tangent at (x,y) cannot lie along the back-projection of the mirror axis. This follows from the requirement that the mirror axis be orthogonal to the tangent at (x,y) in the projected plane.

Case $s(x,y) = y/x$, $\kappa > 0$: Unique ellipse centered at the origin. The parameterization, $x = a\cos(t - t_0)$ and $y = b\sin(t - t_0)$, easily verifies (11).

Case $s(x,y) = y/x$, $\kappa < 0$: Unique hyperbola centered at the origin. The parameterization, $x = a\cosh(t - t_0)$ and $y = b\sinh(t - t_0)$, easily verifies (11).

Case $s(x,y) = k$, $\kappa \neq 0$: Unique parabola with

[3] We may rewrite the third order differential equation as a system of first order differential equations, $y' = f(x, y_1, y_2, y_3) = [y_2, y_3, 3y_3^2/(y_2 - s(x,y_1))]$ where $y = [y_1, y_2, y_3] = [y, y', y'']$. The primes denote differentiation with respect to x. Now note that $(dy/dx - s(x,y)) \neq 0$ because if the slope of the tangent at (x,y) were the same as that of the back-projection of the mirror axis through (x,y), then their projections could not be at right angles for any viewpoint. It easily follows that f satisfies the Lipschitz condition (see [Coddington, 1961]) and hence the initial value problem, $y' = f(x, y)$ with $y(x_0) = y_0$, has at most one solution on any interval containing x_0.

axis having slope k. The parameterization, $y = t$ and $x = t^2$ with $k = 0$, easily verifies (11).

Hence, we see that segments of conic-sections are the only non-zero curvature solutions to the differential equation whose satisfaction in the neighborhood of (x,y) is a necessary condition for local bilateral symmetry to be exhibited by an analytic curve[4]. As an aside, straight-lines are the zero curvature solutions to the equation. The complete curve is uniquely specified by analytic extension. Now note that orthographic projections of ellipses, parabolas and hyperbolas always belong to the corresponding family[5]. Hence, conic-section segments do project to curves exhibiting local bilateral symmetry. It follows that, besides straight-lines and pairs of straight-lines meeting at a point, segments of conic-sections are the only planar curves which may exhibit local bilateral symmetry under orthographic projection with a *general* viewpoint.

Q.E.D.

Acknowledgement

This work was inevitably influenced by Tom Binford, who introduced the author to computer vision. Raz Stowe suggested that I formulate my arguments towards the proof of the theorem in the Appendix using a third order differential equation. Discussions with Hong-Seh Lim prompted the generalization of Corollary 2 in the Appendix to the Theorem. This work was supported in part by the Defense Advanced Research Projects Agency under contract N00039-84-C-0211.

[4] Instead of formulating the third order differential equation we could equivalently have presented a procedure which would result in a unique numerical solution for the curve, given the position, tangent and curvature at (x,y). Given this information at (x,y), one could first predict, using the Taylor series, the positions and tangents at the points with x-coordinates $x_1 = x \pm \epsilon$. Then, as $s(.,.)$ determines the back-projection of the mirror axis through every point on the curve, we could predict the position and tangent at the back-projections of the mirror images of (x,y) about the mirror axes through $(x_1,.)$. The mirror axes through these newly determined points could now be used to further extend the curve, and so on. The errors in the positions and tangents of $(x_1,.)$ are proportional to ϵ^3 and ϵ^2, respectively, and the number of iterations needed to describe the curve-segment is proportional to $1/\epsilon$. Hence, as we let $\epsilon \rightarrow 0$, the numerical solution will converge to the true solution.

[5] This can be verified for conic-sections by noting that the discriminant, $b^2 - 4ac$, of the quadratic, $ax^2 + bxy + cy^2 + dx + ey + f = 0$, does not change sign under orthographic projection.

References

Attneave, F., and Frost, R., "The determination of perceived tridimensional orientation by minimum criteria," *Perception and Psychophysics*, Vol. 6 (6B), 1969, 391-396.

Coddington, E.A., *An Introduction to Ordinary Differential Equations*, Prentice-Hall, New Jersey, 1961.

Coxeter, H.S.M., *Introduction to Geometry*, John Wiley & Sons, New York, 1961.

Guillemin, V., and Pollack, A., *Differential Topology*, Prentice-Hall, New Jersey, 1974.

Hilbert, D., and Cohn-Vossen, S., *Geometry and the Imagination*, Chelsea, New York, 1952.

Kanade, T., "Recovery of the Three-Dimensional Shape of an Object from a Single View," *Artificial Intelligence* 17, 1981, 409-460.

Koenderink, J.J., and van Doorn, A.J., "The shape of smooth objects and the way contours end," *Perception*, Vol. 11, 1982, 129-137.

Koenderink, J.J., "What does the occluding contour tell us about solid shape?," *Perception*, Vol. 13, 1984, 321-330.

Lipschutz, M.M., *Differential Geometry*, McGraw-Hill, New York, 1969.

Lowe, D.G., and Binford, T.O., "The Recovery of Three-Dimensional Structure from Image Curves," *IEEE Trans. PAMI-7*, No. 3, May 1985, 320-326.

Malik, J., "Interpreting Line Drawings of Curved Objects," Doctoral Dissertation, Computer Science Department, Stanford, 1985.

Marr, D., "Analysis of Occluding Contour," *Proc. R. Soc. Lond. B.* 197, 1977, 441-475.

Nalwa, V.S., "Line-Drawing Interpretation : Straight-Lines and Conic-Sections," *Proc: Image Understanding Workshop*, Los Angeles, Feb. 1987.

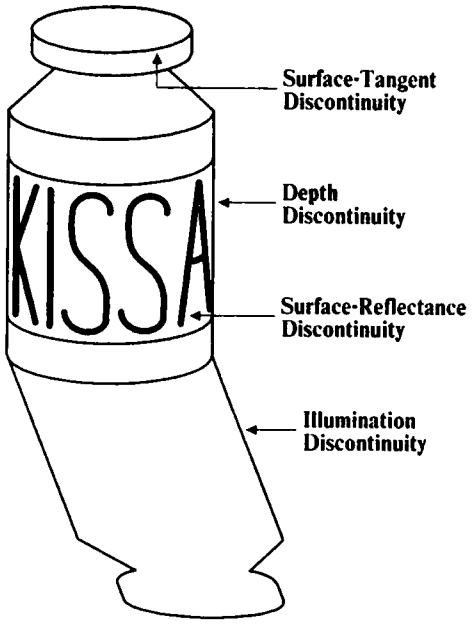


Fig. 1. Line-Drawing with Various Types of Edges

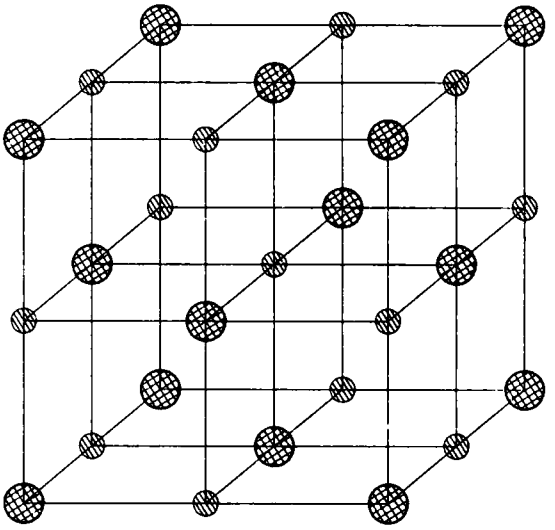


Fig. 2-a. Crystals Exhibit a Symmetric Atomic Arrangement (The Sodium Chloride Crystal Structure is Shown)

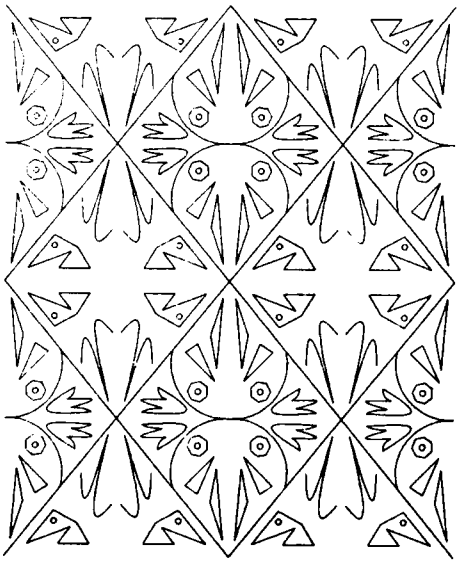


Fig. 2-b. Wallpaper Pattern Exhibiting Various Symmetries

Fig. 2-c. A Chair Exhibits Reflective Symmetry in Space

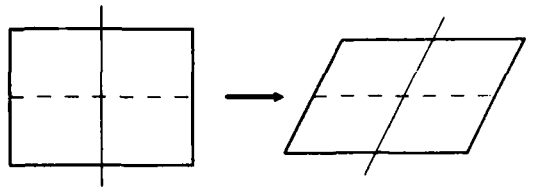


Fig. 3. Bilateral Symmetry in a Planar Figure Maps to Skewed Symmetry under Orthographic Projection

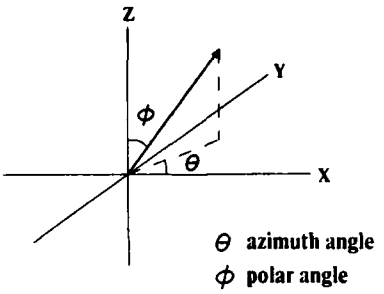


Fig. 4. The Coordinate Frame

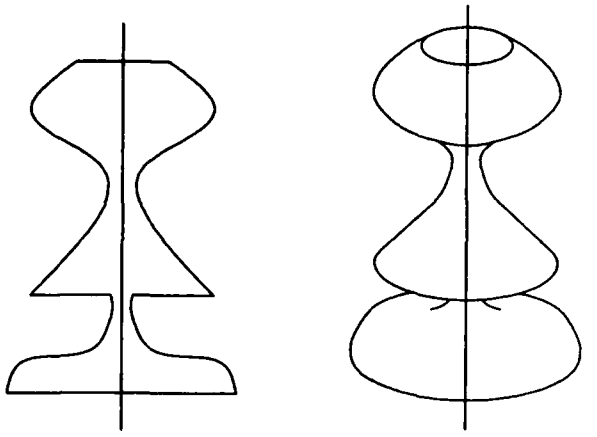


Fig. 5. A Surface of Revolution

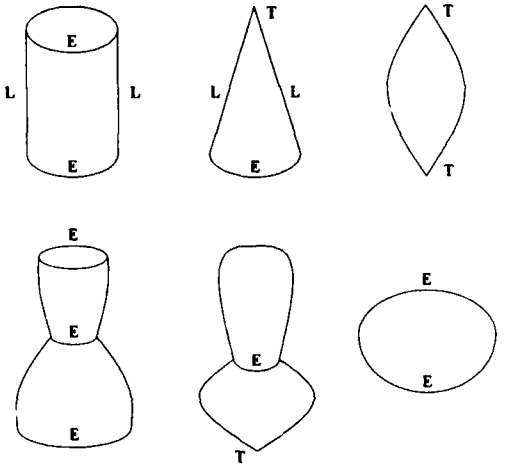


Fig. 6. Examples of Bilaterally Symmetric Line-Drawings from which Local Surfaces of Revolution may be Deduced (Cues : L - Line, E - Ellipse, T - Tip)

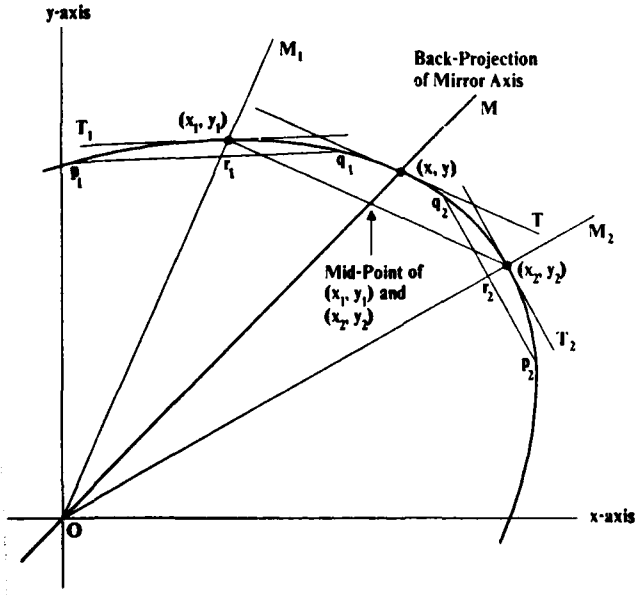


Fig. 7. The Back-Projections of the Mirror-Axes must Intersect at a Single Point

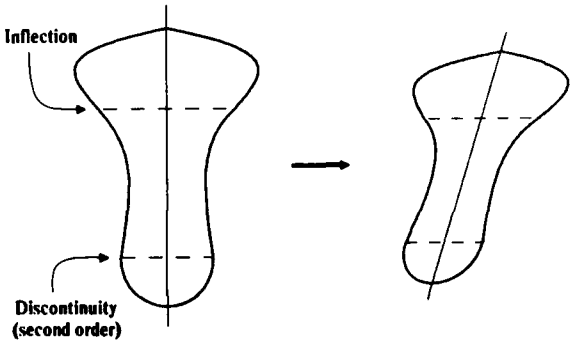


Fig. 8. Bilateral Symmetry between Pairs of Inflections and Discontinuities Maps to Skewed Symmetry between them under Orthographic Projection

ESTIMATION AND ACCURATE LOCALIZATION OF EDGES [1]

Fatih Ulupınar and Gérard Medioni

Institute for Robotics and Intelligent Systems
Departments of Electrical Engineering and Computer Science
University of Southern California
Los Angeles, California 90089-0273

Abstract

The Laplacian of Gaussian (LoG) operator is one of the most popular operators used in edge detection. This operator, however, has some problems: all zerocrossings do not correspond to edges and also, zerocrossings do not always accurately localize edges. In this paper, we offer solutions to these two problems for one dimensional signals, such as slices from images. For the problem of bias, we propose different techniques; the first one combines the results of the convolution of two LoG operators of different standard deviations, whereas the others sample the convolution with a single LoG filter at two points besides the zerocrossing. In addition to localization, these methods allow us to further characterize the *shape* of the edge. We also discuss the problem of edge interaction and offer a partial solution for it. Results on synthetic and real images are presented.

1 Introduction

Processing of visual information begins with the transformation of a very large amount of *viewer centered* visual input into a set of meaningful *object centered* description. Physical edges are one of the most important properties of an object. In the projection transform onto a retina, these edges are very likely to create intensity changes. The reverse is not true, as an intensity change can have its source in a change of some property of an illuminated surface, such as reflectance or surface marking. Therefore it is reasonable to assume that the very first step in the analysis of an image is the detection of intensity changes.

It can be argued that just the *position* of edges is a strong indication of shape, as they allow understanding and recognition of objects in cartoons. We believe that success at subsequent levels in a vision system crucially depends on good features extracted from the signal. These extracted features should have the properties of *completeness*, that is they should capture all of the information present in the intensity image, and of *meaningfulness*, that is they should make the important object centered properties explicit.

Since the work of Roberts [15] in the early 60's, a number of edge detectors have been proposed. They can be classified in two broad classes, depending on whether they use first or second derivative properties.

1.1 First Derivative Edge Detectors

In this approach, edges can be detected as local maxima of the image convolved with a first derivative operator. Roberts [15] and Sobel [3,5] use a straightforward implementation of this idea. Differentiation, however, is a noise enhancing operation, forcing researchers to perform a smoothing step prior to differentiation, as described in [5].

Recently, Canny [2] derived the "optimal" shape of such an operator, based on signal to noise ratio considerations; it is a linear combination of four exponentials which is closely approximated by the first derivative of a Gaussian. This edge detector performs rather well on real images and has been used by many authors. However, the biasing problem that we discuss in this paper also affects this operator since it uses a Gaussian function as a smoothing filter.

It should be noted that all detectors using first derivative require a *thinning* step, since an edge responds with a broad peak.

1.2 Second Derivative Edge Detectors

Here, edges are detected as the location where the second derivative of the (smoothed) image crosses zero. One of the widely used operators in this class has been proposed by Marr and Hildreth [1], and, as a result, the performance of most edge detectors is compared to it. The basic idea is to convolve the image with a Laplacian of Gaussian mask. The

[1]This research was supported by the Defense Advanced Research Projects Agency under contract number F33615-84-K-1404, monitored by the Air Force Wright Aeronautical Laboratories, Darpa Order No. 3119.

amount of smoothing is controlled by the variance of the Gaussian. The main objection to the use of this operator has been that it has poor localization properties and introduces a bias in the edge location estimation, see [6,8,9,14].

Another operator of interest has been proposed by Haralick [8], where, edges are detected as the zerocrossings of the second *directional* derivative in the direction of the gradient. Derivatives are computed by fitting a facet model to the sampled intensity values. On some images, the resulting edges are visually better than the ones from the Marr-Hildreth detector. But this method is computationally more expensive than the Laplacian of Gaussian operator. Torre and Poggio [7] give an excellent analysis of the difference between the Laplacian and the second derivative in the direction of the gradient.

Also worth noting is the work of Nalwa and Binford [6], in which a Tanh curve is fit to the signal. This operator is not linear and does not lend itself to a multiscale approach, but claims to reduce the error in the estimation of the location of an edge.

In the study presented here, we will demonstrate that the bias introduced by a LoG filter can be corrected, and we will show different methods to perform this correction, with the common characteristic that they all perform only a few local operations.

The next section presents an analysis of the zerocrossings obtained after convolution with a LoG operator. We first show that some zerocrossings do not correspond to edges, then express the bias in position as a function of the parameters of an edge. Section 3 presents how this bias can be corrected using two different scales, section 4 presents two different methods to correct bias using a single scale. In section 5, we present another source of error in edge position and give a partial solution to it. Finally, we summarize our results and outline future research.

2 Analysis of the Zerocrossings from a LoG Operator

The Laplacian of Gaussian operator, introduced by Marr and Hildreth [1], uses a Gaussian function to filter out the noise component of the input signal and takes the laplacian of the resulting smoothed signal. In practice, both of the above processes are performed in one step by convolving the input signal with a Laplacian of Gaussian, the LoG operator. The LoG operator has many useful properties, which makes the operator interesting, such as: the effect of noise can be reduced by changing the standard deviation (σ) of the operator. The level of the resolution of the operator can be changed by manipulating the σ. Many studies show that, the Gaussian function is a very good approximation for an optimal filter to be used in edge detection, see [2,7]. In addition, the convolution can be performed very efficiently, either by approximating the LoG by a DoG (Difference of Gaussians) [1], or by other decompositions [10,11]. The LoG operator has many useful properties, but it also has some problems:

- Zerocrossings do NOT always correspond to actual edges, but to inflection points instead.
- Zerocrossings do not always indicate the true position of an edge.

2.1 False Response of LoG Operator

Second derivative operators generate a zerocrossing whenever the input function has an inflection point. This inflection point generally corresponds to step edges in the input signal but not always. Figure 1 shows such a case, in which the middle zerocrossing does not correspond to any step edge. In fact, it results from an inflection point with a tangent parallel to the x-axis, and therefore has a small first derivative. An easy solution for this problem would be to apply a threshold to the value of the first derivative as proposed by Haralick[8]. Thresholding, however, is an ill posed problem, as there is no guarantee to find the right threshold for any input.

Here we propose a method which does not require any thresholding. We define a step edge as an inflection point with a positive maximum or negative minimum of the first derivative. This translates into a simple test:

A zerocrossing correspond to an edge iff

$$f' f''' < 0$$

Note that we do not have to really compute f''', but instead $sgn(f''')$, which is nothing but the polarity of the zerocrossing of the output of LoG operator. For the calculation of f' we have to convolve the input curve with the first derivative of gaussian of the same σ as used in convolution with second derivative of gaussian. Note that we do not have to do this at every point, just at the location of the zerocrossing.

2.2 Bias Introduced by a LoG Operator

In the literature it has been pointed out repeatedly by [8,9] and [14], that a LoG operator has poor localization as σ grows, in fact it biases the original position of the edge. To analyze this bias, let us convolve the operator with the following function:

$$I(x) = \begin{cases} k_1 x & if\ x \le t \\ k_2 x + d & otherwise \end{cases} \quad (1)$$

This defines a step edge at location $x = t$, the height of the step is d and the slopes either sides of the edge are k_1 and k_2. The result of the convolution with a LoG operator is (note that in 1-D, the LoG operator is nothing but second derivative of gaussian function):

$$(I * G'')(x) = [\frac{d(x-t)}{\sigma^2} - (k_2 - k_1)]G(x-t) \quad (2)$$

where G(u) is the gaussian function defined as (neglecting the normalizing constant $\frac{1}{\sqrt{2\pi}\sigma}$)

$$G(u) = e^{\frac{-u^2}{2\sigma^2}} \quad (3)$$

The zerocrossing of equation 2 is located at

$$x_z = t + [\frac{k_2 - k_1}{d}]\sigma^2 \quad (4)$$

Therefore the bias is

$$x_z - t = [\frac{k_2 - k_1}{d}]\sigma^2 \quad (5)$$

This is shown in figure 2, which shows a step edge with different slopes on each side together with the corresponding convolution with a LoG operator. Note the displacement of the zerocrossing with respect to the original location of the step edge.

We present two different methods to correct the bias introduced by the LoG operator. In the first one, we perform the convolution at two different resolutions and obtain t by solving equation 4. In the second one, we use a single resolution and sample the convolved output at more locations besides the zerocrossing. As we go along, we illustrate our approach by showing the processing of two signals, one synthetic and the other a slice from a real image. They are displayed in figure 3 (real image) and 4 (synthetic). For these, (a) shows the intensity image obtained by replicating a slice, (b) shows the intensity variation along any of the rows of the image, and (c) shows the displacement of zerocrossing (-), negative extrema (- ·), and positive extrema (- -) as a function of the filter variance σ.

3 Bias Correction Using Multiresolution

In this method, the input signal is convolved with two operators of different σ values to find the true location of the step edge detected by a zerocrossing. Equation 4 can be rewritten as :

$$x_z = t + c\sigma^2 \quad (6)$$

where

$$c = \frac{k_2 - k_1}{d} \quad (7)$$

In equation 6, there are two unknowns, t, the location of the step edge and c, the ratio of slope difference on each sides of the edge to the height of the edge. So, we need two independent equations, or two samples of the same equation (equation 6) to solve for these unknowns. The method of multiresolution uses convolution outputs of two LoG operators with different σ values, say σ_1 and σ_2. Then we have two linear equations in t, and c.

$$\begin{cases} x_{z_1} = t + c\sigma_1^2 \\ x_{z_2} = t + c\sigma_2^2 \end{cases} \quad (8)$$

and the straightforward solutions for t and c are:

$$c = \frac{x_{z_2} - x_{z_1}}{\sigma_2^2 - \sigma_1^2} \quad (9)$$

$$t = x_{z_1} + \frac{x_{z_2} - x_{z_1}}{\sigma_2^2 - \sigma_1^2}\sigma_1^2 \quad (10)$$

With these we do not only find the actual location of the step edge, t, but we can predict the movement of a zerocrossing as σ changes, since equation 6 is completely solved.

This method applied to the signal in figures 3 and 4 is shown in figures 5 and 6. Figure 5-6 (a) shows the corrected position ($\sigma = 0$) extrapolated from convolutions with two different size filters ($\sigma_1 = 3.0$ and $\sigma_2 = 3.5$). Figure 5-6(b) shows further description for each detected edge: position with *subpixel* accuracy, together with the quantity $c = \frac{(k2-k1)}{d}$.

This method has some problems:

First we have to match zerocrossings of two convolution outputs with different σ values. As clearly seen in figure 3(c), zerocrossings may disappear as σ grows. Thus, we may not have a one to one correspondence of zerocrossings in two convolution outputs. The zerocrossings in the convolution with a small σ may correspond to one or no zerocrossing in the convolution with a larger σ. We have to solve a problem of correspondence, which requires some more decision taking, therefore more computation.

The second and more important problem is the interaction between nearby edges. Since the LoG operator has a relatively large support (approximately 7σ pixels), nearby features affect the convolution output around the edge of interest, which leads us to distorted or incorrect results.

4 Bias Correction Using a Single Resolution

The second method to calculate the values t and c, is to use just one convolution output. There are two approaches in using a single resolution. The first one uses the locations of the maximum minimum and zerocrossing to correct the bias. The second method uses the convolution output values at arbitrary locations around the zerocrossing.

4.1 Using Maximum and Minimum

This method utilizes the locations of the maximum, minimum and zerocrossings associated with a step edge. Before going into the details of the method, let us derive the equations for the locations of the maximum and the minimum of the convolution output for the same step edge model. The location of the maximum is:

$$x_p = [\frac{c\sigma^2 + \sigma\sqrt{(c^2\sigma^2+4)}}{2}] + t \qquad (11)$$

and the location of the minimum is :

$$x_n = [\frac{c\sigma^2 - \sigma\sqrt{(c^2\sigma^2+4)}}{2}] + t \qquad (12)$$

where c is given by equation 7. When we sum the locations of the extremes (x_p and x_n) and subtract the location of the zerocrossing we get the true location of the step edge, t, a very elegant invariant of the problem.

$$\boxed{t = x_p + x_n - x_z} \qquad (13)$$

where x_z is the location of the zerocrossing in the convolution output, and x_p , x_n are given above.

The location of the step edge is not the only information that we can get out of these formulas. We can also obtain the height of the step edge, d, and the slope difference at each side of the step edge, $(k_2 - k_1)$. The equation for the height of the step edge can be obtained from equation 2:

$$d = \frac{m\sigma^2}{l} e^{\frac{(x_z - t + l)^2}{2\sigma^2}} \qquad (14)$$

where $m = (I * G)(x_z + l)$. That is, m is the value of the convolution at location $x_z + l$ for some arbitrary l. Also the slope differences can be obtained from the equations 6 and 7, given as:

$$(k_2 - k_1) = \frac{x_z - t}{\sigma^2} d \qquad (15)$$

As presented here, we do not only get the actual location of the step edges with subpixel accuracy but we can also get much more information about the step edge; the height and the slope difference on each side, from the convolution with a LoG filter. In particular, the true height of the edge can be used in thresholding the output of the operator since it is usually a problem to find a natural thresholding criteria in second derivative operators.

In figure 7 and 8 the result of this method applied to the signals in figure 3 and 4 is presented. With this method, as in the previous one, we can also predict the movement of the zerocrossing and the extrema of the convolution output as σ changes. Figures 7-8 (a) show the predicted position of extreme and zerocrossing based on their actual position in the convolution with a LoG filter of variance σ (here σ = 3.0). Figures 7-8(b) list each edge location together with its estimated step size (d), and slopes difference $(k_2 - k_1)$, and the amount of the correction in localization $(x_z - t)$.

4.2 Second Approach to Single Resolution

Further analysis of the equations show that we do not need to find the location of the maximum and the minimum in the convolution output to find t. If we sample the convolution output at two different locations say at $x_z + l_1$ and at $x_z + l_2$, and if the values obtained at these samples are m_1 and m_2, that is

$$m_1 = [\frac{d(x - t + l_1)}{\sigma^2} - (k_2 - k_1)]G(x - t + l_1) \qquad (16)$$

and

$$m_2 = [\frac{d(x - t + l_2)}{\sigma^2} - (k_2 - k_1)]G(x - t + l_2) \qquad (17)$$

The ratio of these two expressions gives us:

$$\frac{m_1}{m_2} = [\frac{x_z - t + l_1 - c\sigma^2}{x_z - t + l_2 - c\sigma^2}] \exp(\frac{-(x_z - t + l_1)^2 + (x_z - t + l_2)^2}{2\sigma^2}) \qquad (18)$$

where c is given in equation 7. Also note that $c\sigma^2$ is equal to $x_z - t$, given in equation 6. With this substitution we get:

$$\frac{m_1}{m_2} = \frac{l_1}{l_2} \exp(\frac{l_2^2 - l_1^2 + 2(x_z - t)(l_2 - l_1)}{2\sigma^2}) \qquad (19)$$

This equation is analytically solvable in t

$$t = x_z - \frac{1}{2(l_2 - l_1)}(2\sigma^2 log\left|\frac{m_1 l_2}{m_2 l_1}\right| - l_2^2 + l_1^2) \qquad (20)$$

In the above equation if we set $l_2 = -l_1$, we get a much simplified equation, but when working with discrete data one will need to interpolate the values m_1 and m_2, with this restriction on l values. The simplified equation is :

$$t = x_z + \frac{\sigma^2}{2l_1} log\left|\frac{m_1}{m_2}\right| \qquad (21)$$

For choosing the values l_1 and l_2 there are two considerations. First, the smaller these values are the equation will be singular and ill posed and the accuracy will be poor. Second, the larger these values are the problem of interaction will be more severe, which is discussed in the next section. It is a trade off between these two. In the ideal case, there is no distorsion and no interaction, all values should produce the same result.

This medthod applied to the signals in figures 3 and 4 are shown in figures 9 and 10. Figures 9-10 (a) show the predicted positions of extrema and zerocrossing based on their actual position in the convolution with a LoG filter of variance σ (here $\sigma = 3.0$), and ($l_1 \approx 1.5$). Figures 9-10(b) list each edge location together with its estimated step size and slopes difference, and the amount of the correction in localization.

4.3 Problems with the Model

Both of the above methods, the multiresolution and single resolution methods, work perfectly unless we have the problem of interference between nearby features, which displaces the locations of extrema and zerocrossings, thus the accuracy of the calculations. The multiresolution method seems to work better in the case of the interaction of nearby features, however both methods are affected. The movement of zerocrossing with respect to σ can result either $k_2 \neq k_1$ for that edge or from the interference of nearby features. The multiresolution method samples the *actual* movement of zerocrossings with respect to σ, therefore it also partially corrects the effect of interaction, whereas the single resolution method works on one sample and tries to solve the problem without knowing the actual movement of the zerocrossing. One should note, however, that it reduces noticeably the bias due to interaction in a bump, which is studied in the next section.

5 Solving the Interaction Problem

The problem of interaction of nearby features is also studied. The first solution proposes to use a model which consists of two features, in other words, a model which expresses the relation of interaction between two features. The subsequent sections discusses such a model which we call a "bump" model.

5.1 The Bump Model

To solve the interaction problem, a new model, which we call a bump, is used which has the following functional representation :

$$B(x) = \begin{cases} d & if \ -a \leq x \leq a \\ 0 & otherwise \end{cases} \tag{22}$$

and the convolution output for this model is

$$(B * G'')(x) = [\frac{(-(dx - ad)e^{\frac{2ax}{\sigma^2}} + dx + ad)e^{-\frac{x^2+2ax+a^2}{2\sigma^2}}}{\sigma^2}] \tag{23}$$

In figure 11, a typical bump and the resulting convolution output is shown. The locations of the zerocrossings are:

$$x_z = \frac{ae^{\frac{2ax_z}{\sigma^2}} + a}{e^{\frac{2ax_z}{\sigma^2}} - 1} \tag{24}$$

Note that the solution for zerocrossings given above is not an explicit solution in terms of x_z. Unfortunately, there is no explicit analytical solution for that formula. Since there are no analytical solutions for the above formula, it is solved using the Newton-Rapson method. From that equation, we can find the a value (half of the width of the bump), and the location of the bump is just in the middle of the two zerocrossing that we get from the convolution. In addition, we are able to find the height of the bump by the following equation:

$$d = \frac{m\sigma^2 e^{\frac{x_m^2+2ax_m+a^2}{2*\sigma^2}}}{(-(x_m - a)e^{\frac{2ax}{\sigma^2}} + x_m + a)} \tag{25}$$

where x_m is the local extremum between zerocrossings and m is the magnitude of convolution at location x_m.

This bump model is in fact not a solution to the problem of interaction of nearby features, as it creates more problems than it solves: First, it is a very unflexible model, as it restricts the two sides of the bump to be equal, which is rarely the case in real images of course. The second problem is that this model predicts *two* edge locations for each zerocrossing, if we have two bumps close to each other. The third problem is that it only considers the interaction of two step edges, whereas in the real world we have interaction of more than two edges (practically, all edges closer than the distance 3σ). The first problem can be solved by using a more flexible model as:

$$B(x) = \begin{cases} d_2 & if \ t - a \leq x \leq t + a \\ d_1 + d_2 & if \ x > t + a \\ 0 & otherwise \end{cases} \tag{26}$$

This is a bump with different heights on each side, d_2 on left side and $-d_1$ on the right side. The convolution output for this model is:

$$\frac{(d_1((x - t) - a)e^{\frac{2a(x-t)}{\sigma^2}} + d_2((x - t) + a))e^{-\frac{(x-t)^2+2a(x-t)+a^2}{2\sigma^2}}}{\sigma^2} \tag{27}$$

And the equation for zerocrossing is

$$x_z = a\frac{d_1 e^{\frac{2az}{\sigma^2}} - d_2}{d_1 e^{\frac{2az}{\sigma^2}} + d_2} + t \qquad (28)$$

This is not an explicit solution either. In fact, it is not possible to solve this equation in this form since there are many unknowns (a, d1, d2 and t which is the location of the middle of the bump). Therefore applying this model is no good either. Consequently, the bump model does not give any help in solving the problem of interaction. We believe that the problem of edge interaction is later solved by Chen and Medioni [11].

6 Conclusion and Future Work

Here we have presented a method to refine the position of edges after the detection of the zero crossings of the original signal convolved with a LoG operator. We also estimate the shape of each edge, including the height and the difference of slopes on each side.

As stated in section 4.3 the main disadvantage of this technique is that it does not solve the edge interaction problem, which is studied and solved in [11]. That study, however, does not deal with the bias introduced by the slope differences of the step edge. We believe that a combination of these two methods will produce much better results.

The second step is the extension to 2-D. One solution is to perform convolution in rows and columns of the image and add the results. Theoretically having the information about the derivative of a 2-D surface in two independent directions is sufficient to know the derivative in all directions. This is true when noise is neglected. Still, this method is quite sufficient for finding edges in an image. But the true application to 2-D is achieved when the equations are modeled in 2-D and solved in 2-D. However, these equations may be practically too difficult to solve as they are solved in 1-D slices.

References

[1] Marr, D. C. and Hildreth, E. C., "Theory of Edge Detection," Proceedings of the Royal Society of London, B207, 1980, pp 187-217.

[2] Canny, F. R., "Finding Edges and Lines in Images," Technical Report 720, MIT AI Lab., June 1983.

[3] Nevatia, R., "Machine Perception," Princeton Hall, 1982.

[4] Nevatia, R. and Babu, K. R., "Linear Feature Extraction and Description," Computer Graphics and Image Processing, Vol. 13, 1980 pp 257-269.

[5] Rosenfeld A. and Kak A. C., "Digital Picture Processing," Academic Press, 1982, Vol. 2.

[6] Nalwa, V. S. and Binford, T. O., "On Detecting Edges," Proceedings of Image Understanding Workshop, Miami, December 1985, pp 450-465.

[7] Torre, V. and Poggio, T. A., "On Edge Detection," IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. 8, No. 2, March 1986, pp 147-163.

[8] Haralick, R. M., "Digital Step Edges from Zerocrossings of Second Directional Derivatives," IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. 6, No. 1, January 1984, pp 58-68.

[9] Berzins, V., "Accuracy of Laplacian Edge Detectors," Computer Vision, Graphics and Image Processing, Vol. 27, 1984, pp 195-210.

[10] Huertas, A. and Medioni G., "Detection of Intensity Changes with Subpixel Accuracy Using Laplacian-Gaussian Mask," IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. 8, No. 5, September 1985, pp 651-664.

[11] Chen, J. S. and Medioni, G., "Detection, Localization and Estimation of Edges," Intelligent Systems Group, USC, Internal Report, 1986, to be published.

[12] Chen, J. S., Huertas, A. and Medioni, G., "Very Fast Convolution with Laplacian-of-Gaussian Mask," Proceedings CVPR-86, Miami, June 1986, pp 293-298.

[13] Watson, L., et al "Topographic Classification of Digital Image Intensity Surfaces Using Generalized Spline and the Discrete Cosine Transformation," Computer Vision Graphics and Image Processing, Vol. 29, 1985, pp 143-167.

[14] Shah, M., Sood, A. and Jain, R., "Pulse and Staircase Models for Detecting Edges at Multiple Resolution," Proceedings of the Third Workshop on Computer Vision: Representation and Control, Bellaire, Michigan, October 1985, pp 84-95.

[15] Roberts, L. G., "Machine Perception of Three Dimensional Solids," Optical and Electro Optical Information Processing, ed. Tippett, J. T., et al., Cambridge, Mass., MIT Press, 1965, pp 150-197.

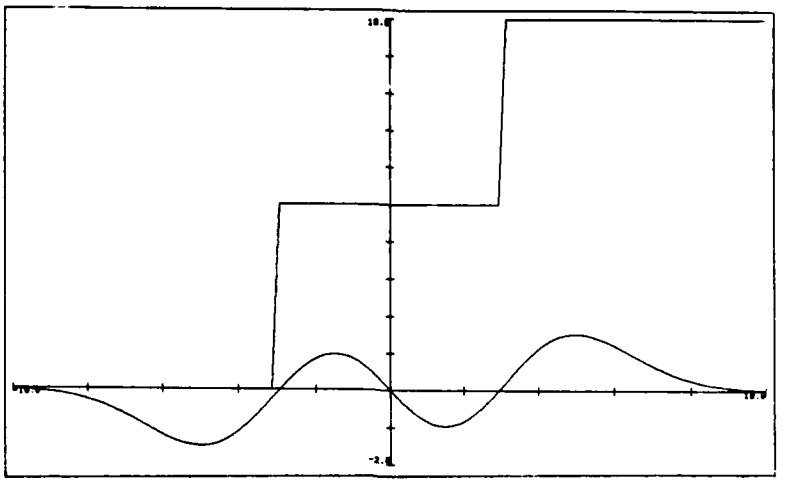


Figure 1: Zerocrossing do not always correspond to edges:
a staircase function and its convolution with a LoG mask ($\sigma = 2.0$).

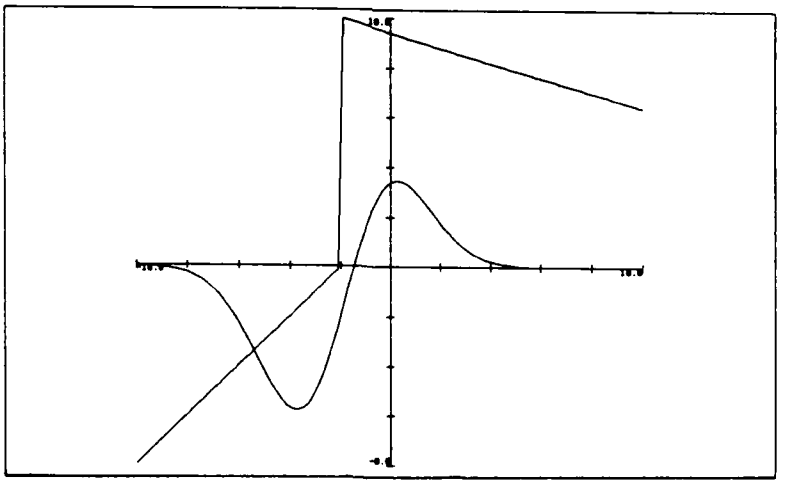
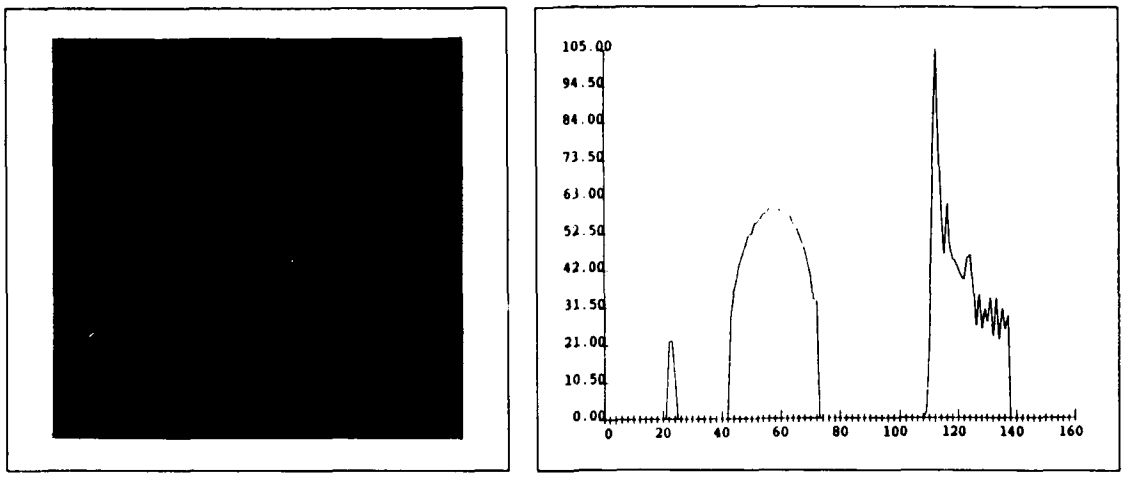
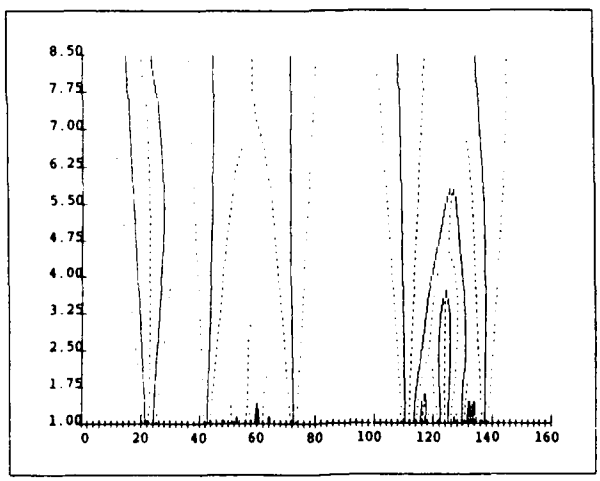


Figure 2: Zerocrossings do not accurately locate edges: a step function ($k_1 = 1.0$, $k_2 = 0.5$, $d = 10$) and its convolution with a LoG mask ($\sigma = 2.0$).

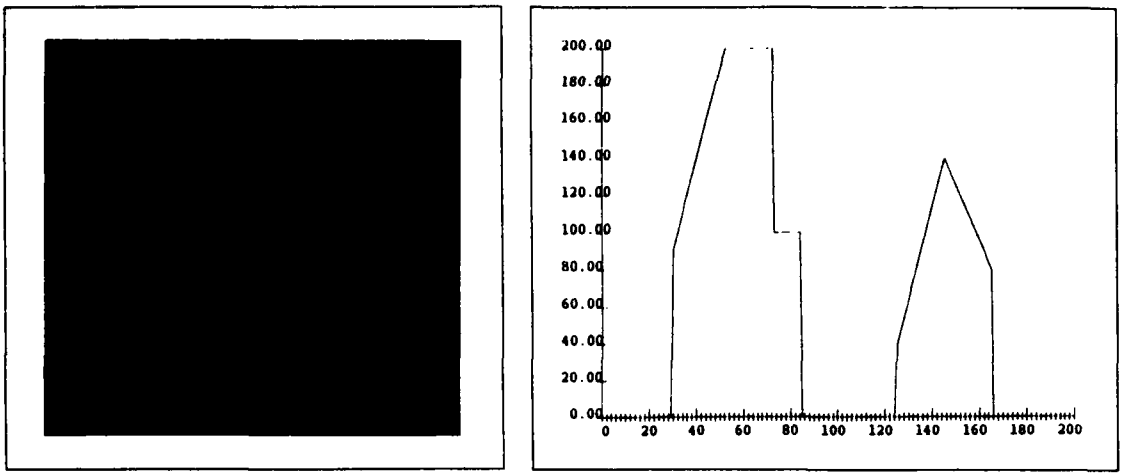


(a) (b)

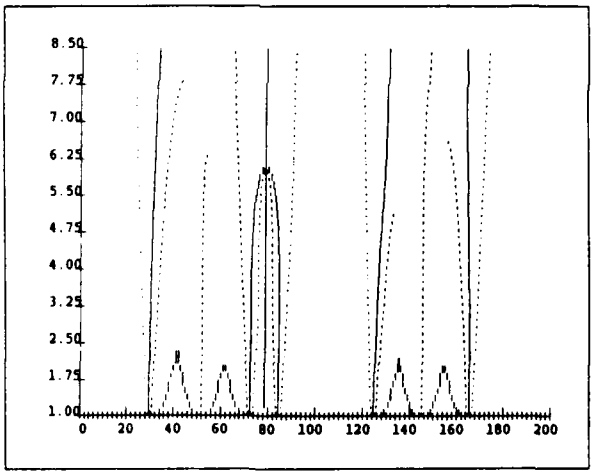


(c)

Figure 3: (a) Intensity image obtained by replicating a slice from a real image. (b) The slice used to produce the image in (a). (c) The displacement of zerocrossing (–), negative extrema (- ·), and positive extrema (- -) in the convolution with a LoG mask as σ changes from 1.0 to 8.5

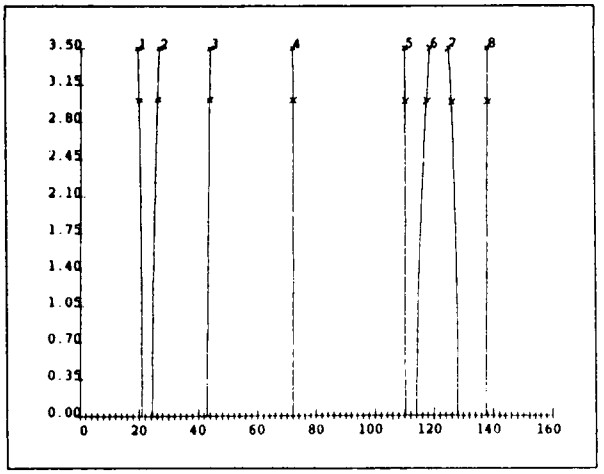


(a) (b)



(c)

Figure 4: (a) Intensity image obtained by replicating a synthetic 1-D signal. (b) The signal used to produce the image in (a). (c) The displacement of zerocrossing (–), negative extrema (- ·), and positive extrema (- -) in the convolution with LoG mask as σ changes from 1.0 to 8.5

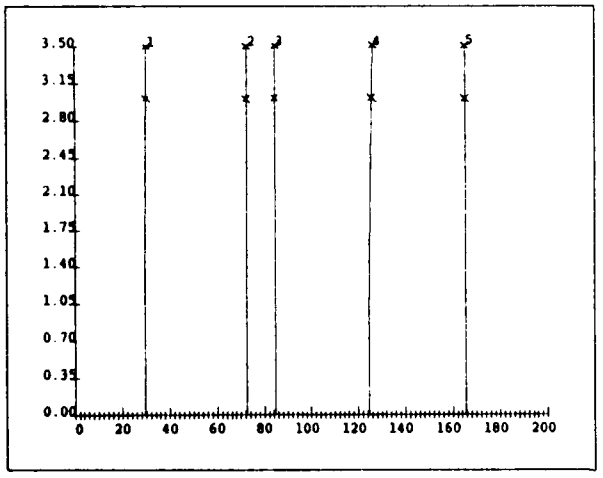


(a)

feature no	location	ratio: (k2-k1)/d
1	20.94	-0.14
2	24.48	0.16
3	43.02	0.05
4	72.03	-0.05
5	110.11	-0.06
6	114.10	0.32
7	127.92	-0.30
8	137.58	-0.02

(b)

Figure 5: Application of the multiresolution method to the real signal in figure 3 with $\sigma_1 = 3.0$ and $\sigma_2 = 3.5$
(a) The predicted $\sigma - x$ space, (b) the listing of features.

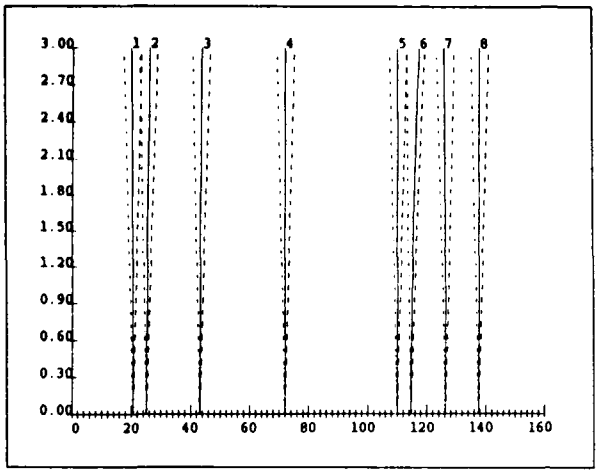


(a)

feature no	location	ratio: (k2-k1)/d
1	29.46	0.06
2	72.42	0.01
3	84.58	-0.01
4	124.52	0.13
5	165.51	-0.04

(b)

Figure 6: Application of the multiresolution method to the synthetic signal in figure 4 with $\sigma_1 = 3.0$ and $\sigma_2 = 3.5$
(a) The predicted $\sigma - x$ space, (b) the listing of features.



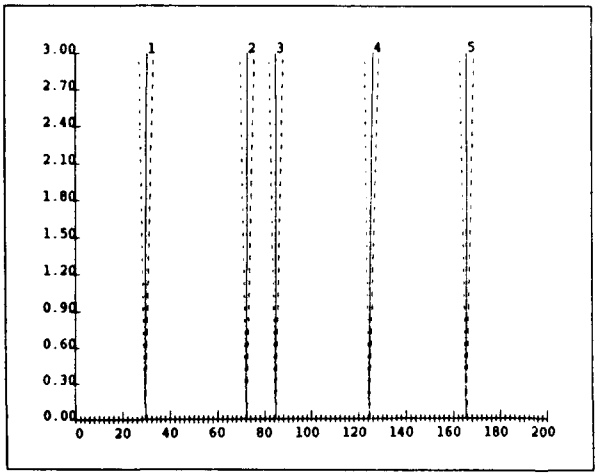
(a)

feature no	location	(k2-k1)	d	correction
1	20.52	-2.13	22.25	0.86
2	25.13	-1.76	-20.59	-0.77
3	43.12	1.60	40.05	-0.36
4	71.92	1.52	-41.40	0.33
5	109.95	-4.29	101.86	0.38
6	114.70	-12.86	-51.46	-2.25
7	126.28	0.86	-7.37	1.05
8	137.59	0.40	-22.45	0.16

(b)

Figure 7: The single resolution method using maxima and minima applied to the signal in figure 3 with $\sigma = 3.0$

(a) The predicted $\sigma - x$ space, (b) the listing of features, showing the height of the step edge and location correction.



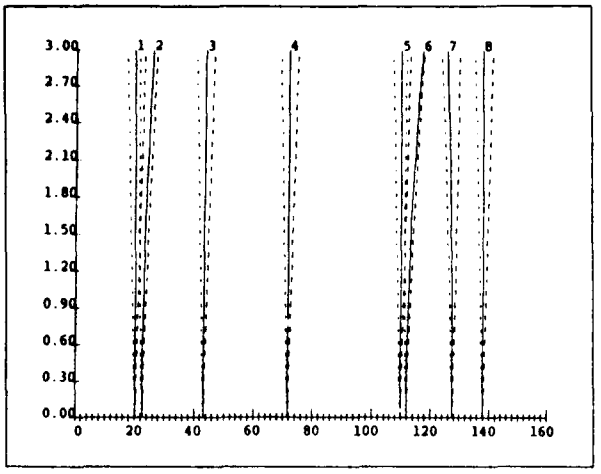
(a)

feature no	location	(k2-k1)	d	correction
1	29.53	4.94	92.94	-0.48
2	72.32	-1.67	-80.06	-0.19
3	84.68	1.91	-91.37	0.19
4	124.50	4.69	35.00	-1.21
5	165.48	3.02	-86.23	0.32

(b)

Figure 8: The single resolution method using maxima and minima applied to the signal in figure 4 with $\sigma = 3.0$

(a) The predicted $\sigma - x$ space, (b) the listing of features, showing the height of the step edge and location correction.



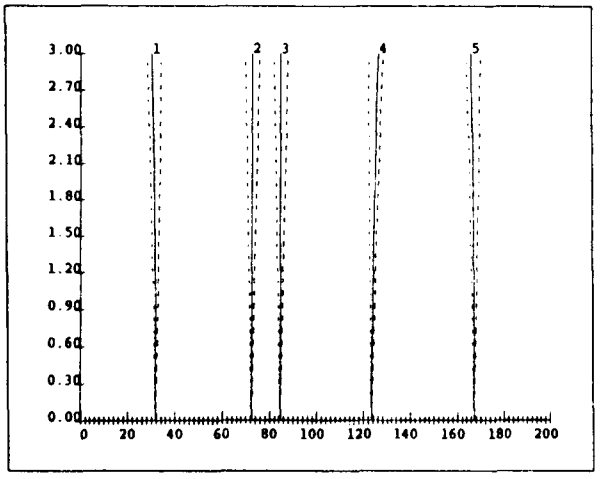
(a)

feature no	location	(k2-k1)	d	correction
1	20.19	-1.44	24.53	0.53
2	22.57	-5.98	-16.15	-3.33
3	43.13	1.54	39.85	-0.35
4	71.61	0.09	-36.39	0.02
5	109.78	-2.55	107.94	0.21
6	111.88	-26.44	-46.95	-5.07
7	127.51	2.77	-10.96	2.28
8	137.91	1.32	-24.74	0.48

(b)

Figure 9: The single resolution method using equation 21 applied to the signal in figure 3 with $\sigma = 3.0$ and $l_1 \approx 1.5$

(a) The predicted $\sigma - x$ space, (b) the listing of features, showing the height of the step edge and location correction.



(a)

feature no	location	(k2-k1)	d	correction
1	29.53	4.94	92.94	-0.48
2	72.32	-1.67	-80.06	-0.19
3	84.68	1.91	-91.37	0.19
4	124.50	4.69	35.00	-1.21
5	165.48	3.02	-86.23	0.32

(b)

Figure 10: The single resolution method using equation 21 applied to the signal in figure 4 with $\sigma = 3.0$ and $l_1 \approx 1.5$

(a) The predicted $\sigma - x$ space, (b) the listing of features, showing the height of the step edge and location correction.

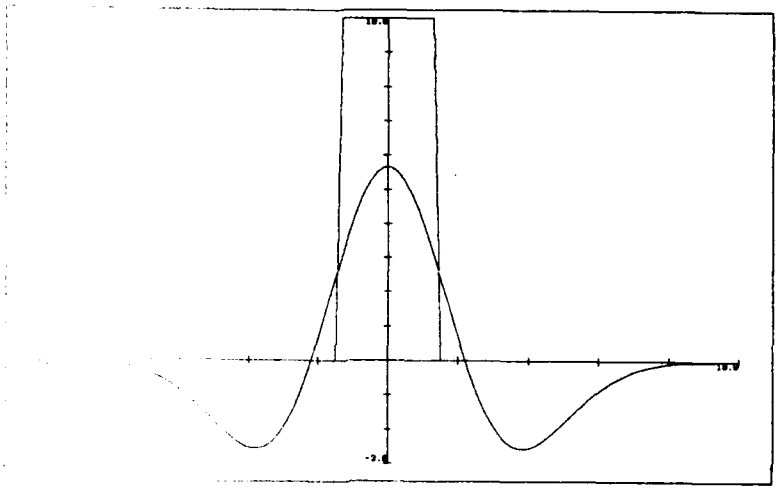


Figure 1: The bump model and its convolution with a LoG mask for

Edge-Detector Resolution Improvement by Image Interpolation

Vishvjit S. Nalwa
A.I. Lab., Stanford University, CA 94305

Abstract

Most step-edge detectors are designed to detect locally straight edge-segments which can be isolated within the operator kernel. While it can easily be demonstrated that a cross-sectional support of at least 4 pixels is required for the unambiguous detection of a step-edge, edges which cannot be isolated within windows having this width can nevertheless be resolved. This is achieved by preceding the step-edge detection process by image-intensity interpolation. Although resolution can be improved in this fashion, the step-edge position and intensity estimates thus determined may be subject to systematic biases. Also, the higher resolution performance is accompanied by lower robustness to noise.

I. Introduction

Most step-edge detectors are designed to detect locally straight edge-segments which can be isolated within the operator kernel (e.g. [8], [5], [9]). It was pointed out in [9] that the minimum lateral support required for the unambiguous detection of a step-edge was 4 pixels. The underlying implication was that step-edges for which such a support was unavailable would require special-purpose detectors, e.g. line-detectors. It is demonstrated here that this is often unnecessary. Contrary to common belief (see [8], [4], [5], [9]), so-called line-edges can be detected by step-edge detectors if we first reconstruct the underlying image-intensity surface.

Line-edges have often been mentioned in literature (see [4]) without, to the best of our knowledge, it ever being explicitly stated when two locally parallel straight step-edges constitute a line-edge, i.e. at what separation between two step-edges do we consider them to constitute a single line-edge, and why? The answer, we claim is system dependent. As discussed at length in [9], edges undergo "blurring" when registered by any imaging device[1]. Whenever this "blur" is sufficient to cause the images of two parallel step-edges to interact, we have a line-edge. This is illustrated in Fig. 1 for the 1-D continuous case. If we assume the blurring function to be well-approximated by a Gaussian, as is often done for simplicity, then we can consider pairs of step-edges separated by less than $5\sigma_{\text{blur}}$ to constitute line-edges. (Typically, $\sigma_{\text{blur}} >= 0.5$ pixel for imaging systems with little aliasing.)

In this paper we seek to demonstrate how so-called line-edges may be detected by a step-edge detector. This is achieved by reconstructing the underlying image-intensity surface. Line-edges need to be discovered nevertheless, not because it is not possible to detect them as a pair of step-edges, but because we want to accurately recover the parameters of the underlying steps. As is evident from Fig. 1, the interaction between the images of the component

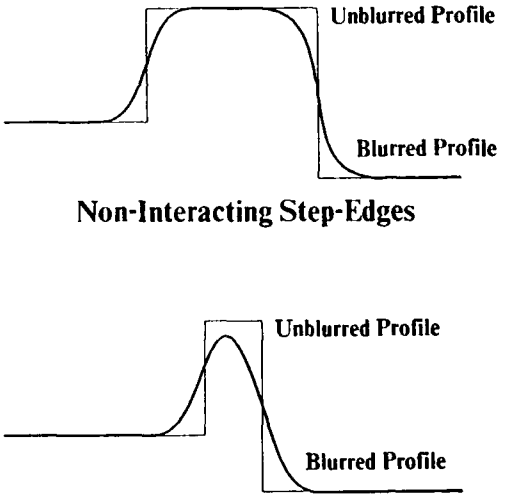


Fig. 1. Non-Interacting and Interacting Step-Edges

This paper has been accepted for publication in *IEEE Transactions on Pattern Analysis and Machine Intelligence*.
This work was supported in part by the Defense Advanced Research Projects Agency under contract N00039-84-C-0211.

[1] There are two fundamental sources for the "blur" associated with any imaging system. One is the diffraction limit (see [6]) and the other is the finite sensor aperture. Optical aberrations and defocus may also contribute (see [6]).

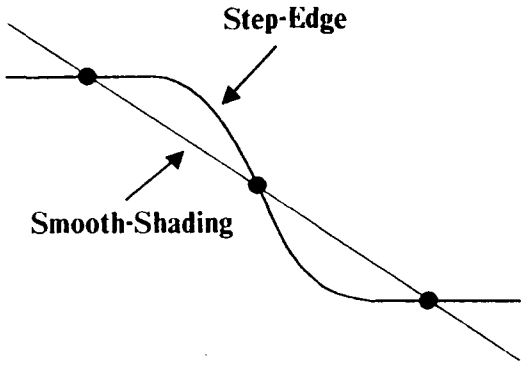


Fig. 2. Ambiguity in profile, given 3 symmetric samples of a step-edge cross-section.

step-edges will result in systematic errors in their position and intensity estimates --- the discovery of line-edges is needed to remove these biases. (In general, non-zero slopes on the two sides of step-edges will also bias the parameter estimates.) We concern ourselves here with the detection aspect and do not address in any detail the issues pertaining to the systematic errors.

As an aside, we mention that it has been speculated in the psychophysical literature that explicit image interpolation is performed by the human visual system. For instance, Barlow [3] has suggested that discrete spatial interpolation between retinal samples may be performed at the terminations of the optic radiation fibers in the cortex.

In Section II the underlying procedure is outlined and in Section III the design of the interpolation filter is discussed. Section IV presents some examples with which we hope to highlight the methodology and demonstrate its performance on "real" images. We conclude in Section V with a discussion.

II. The Procedure

In [9] it was illustrated that 3 symmetric samples were insufficient to detect a 1-D step-edge owing to inherent ambiguities. This is depicted in Fig. 2. It is clear, however, that if we had available the continuous signal itself, then our window size would not be thus restricted, i.e. a 3 pixel window on the continuous signal has no such ambiguity. But, if our signal is by and large bandlimited, then we can use an interpolation filter to reconstruct the underlying continuous signal. This is a reasonable assumption if σ_{blur}, for the effective Gaussian blurring function associated with the imaging system, is no less than 0.5 pixel. Hence, although we do require more than 3 samples to detect a step-edge, it is not necessary that we be able to isolate the step-edge within such a window. We can instead use the samples to reconstruct the continuous signal and then detect step-edges using

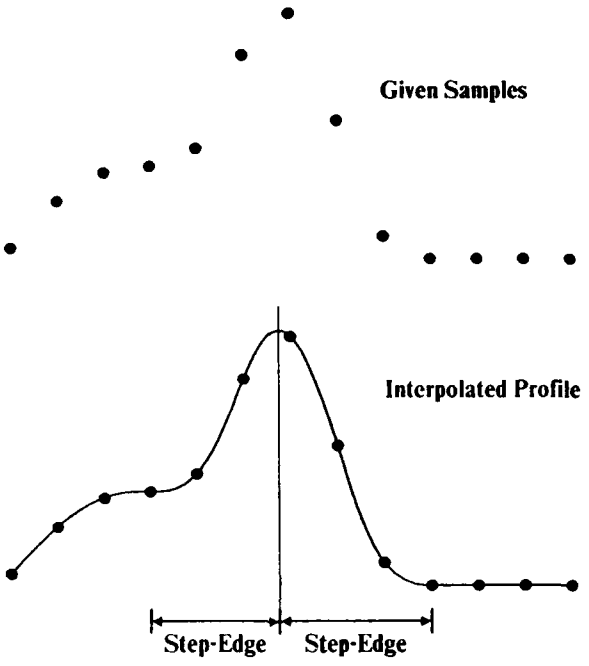


Fig. 3. Procedure for detecting step-edges using interpolation.

smaller windows. This procedure is illustrated in Fig. 3 for the samples of a line-edge. All the foregoing arguments carry over to 2-D. In practice, we need not reconstruct the continuous signal. Interpolation onto a finer grid often suffices. For purposes of demonstration, we chose our new grid to have the half the original inter-pixel spacing.

III. Design of Interpolation Filter

The choice for an interpolation filter which immediately comes to mind for a square sampling grid is sinc(x) sinc(y) where x and y are the row and column axes and $\text{sinc}(x) = \frac{\sin(\pi x)}{\pi x}$ (see [2]). The sinc function, however, is of infinite extent and therefore practically unfeasible. We could window the filter. But then we would be confronted with the choice of the windowing function with its accompanying trade-offs (see [10]). An alternative procedure is to design a filter which has certain desirable characteristics.

Taking the cue from the sinc filter, we can make two immediate observations. First, the 2-D filter can be conceived of as the separable product of two identical 1-D filters. Hence, we need only design a 1-D filter. Second, in order to reconstruct the signal while preserving the given samples we must have unit magnitude at the origin of the filter and zeros at integer-pixel distances from the origin. As mentioned previously, for our purposes it suffices to interpolate the intensities onto a grid with half the original inter-pixel spacing. Hence, we need not design a continuous

$$\frac{1}{256}\begin{bmatrix} 1 & 0 & -9 & -16 & -9 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -9 & 0 & 81 & 144 & 81 & 0 & -9 \\ -16 & 0 & 144 & 256 & 144 & 0 & -16 \\ -9 & 0 & 81 & 144 & 81 & 0 & -9 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & -9 & -16 & -9 & 0 & 1 \end{bmatrix}$$

Fig. 4. Interpolation filter designed to preserve functions belonging to the space spanned by $\{1\ x\ x^2\ x^3\} \otimes \{1\ y\ y^2\ y^3\}$. The sampling grid for this filter has a half-pixel spacing.

filter. It is enough to have its samples at half-pixel spacings. Thus, if we decide on a 1-D filter with a 4 pixel support, its coefficients will be {0 a 0 b 1 c 0 d 0}, where the coefficients are listed at half-pixel spacings about the origin. The unspecified coefficients can be systematically chosen by requiring that functions spanned by a given basis be reconstructed without distortion. We chose $\{1, x, x^2, x^3\}$ to be that basis. This is the lowest order polynomial basis that can represent inflection points which are exhibited by every step-edge. It is clear that if we choose the coefficients to be symmetric about the center, all odd powers of x would be undistorted at the origin. If we further require $2(a + b) = 1$ and $2\left(\frac{9}{4}a + \frac{1}{4}b\right) = 0$ then we will meet our specified criteria. Solving these equations, we get $\frac{1}{16}\{0\ -1\ 0\ 9\ 16\ 9\ 0\ -1\ 0\}$ to be the filter which will produce distortionless reconstruction for the space spanned by $\{1\ x\ x^2\ x^3\}$. It is of interest to compare this filter with the samples of the truncated *sinc function* at half-pixel spacings : $\frac{1}{16}\{0\ -3.4\ 0\ 10.2\ 16\ 10.2\ 0\ -3.4\ 0\}$. The 2-D filter corresponding to the separable product of our 1-D interpolation filter is shown in Fig. 4.

As with any image-operator, we are interested in the filter's noise characteristics. Let us assume, as is generally done to keep the analysis simple, that the noise is independently, identically distributed (i.i.d.) zero-mean Gaussian with variance σ^2_{noise}. Then the variance of the noise in the interpolated intensities will be $0.41\sigma^2_{noise}$ when the interpolated pixel has 4 nearest neighbors on the original grid and $0.64\sigma^2_{noise}$ when it has 2 nearest neighbors on the original grid. The noise in the interpolated image, of course, is neither identically nor independently distributed any more. Further, we should keep in mind that systematic errors in the reconstruction are inevitable when the sampled surface within the interpolation window does not belong to the space spanned by $\{1\ x\ x^2\ x^3\} \otimes \{1\ y\ y^2\ y^3\}$ (see Appendix). In contrast with our finite-extent interpolation filter, the infinite-extent sinc filter has an associated noise variance of σ^2_{noise} (see [2]). However, it reconstructs all adequately bandlimited signals without systematic errors.

Higher order filters can be similarly designed. It must, however, be borne in mind that as the size of the filter increases, so does the basis required to closely approximate the surface within the window.

Before we proceed to the next section, we mention that there is a large body of literature on function interpolation and approximation (see [1] and the references within). Various competing schemes include spline-fitting and surface modeling by least-squares-fit. Least-squares surface-fitting has previously been used for edge-detection (e.g. [7]). However, this implicitly smooths the image data and consequently, as will be discussed at length in Section V, is inappropriate for the detection of line-edges. We have not investigated the trade-offs accompanying the various possible interpolation strategies. Our primary purpose here is to elucidate the usefulness of image interpolation, in general, as a method for resolution improvement of edge-detectors.

IV. Examples

We first demonstrate the proposed algorithm on 1-D line-edges with varying widths. Figs. 5-a, -b and -c exhibit the unblurred profile (thin line), the blurred profile (think line), the original samples (circled dots) which are available for processing, and the interpolated values (uncircled dots) at half-pixel spacings, for step-edges spaced 3, 2 and 1 pixels apart, respectively. In all three cases, the step on the left has a high of 192 and a low of 128 while the step on the right has a high of 192 and a low of 64. The positions of the steps are indicated directly below them in the figures. σ_{blur} for the blurring Gaussian used in all three cases is 0.6 pixel. The estimated step-edge parameters are shown alongside the unblurred steps. The step-edge detector outlined in [9] has been used here. As is evident from the figures, the biases in the position and intensity estimates increase as the spacing between the two step-edges decreases for a given σ_{blur}. It is apparent from the blurred profiles that this trend in the biases is not particular to the chosen edge-detector.

We now illustrate the improvement in performance accompanying the application of the proposed technique to "real" images. Fig. 6-a is the original (128 x 128) image of a bin of parts. Fig. 6-b is the corresponding edge-detector output adapted from [9]. Fig. 6-c is the result of applying the same edge-detector with identical parameters to the interpolated image. It is clear that interpolation onto a finer grid results in a marked improvement in the resolution. Many of the detected edges, e.g. those corresponding to the pins in the top-right of the image, are definitely less than $5\sigma_{blur}$ apart (σ_{blur} was estimated to be 0.6

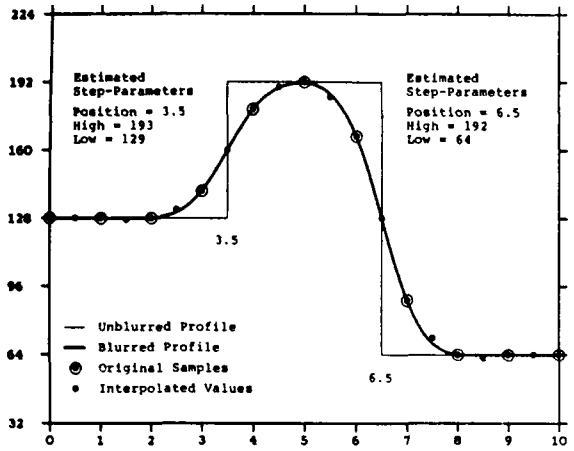


Fig. 5-a. Detection and estimation of step-edges spaced 3 pixels apart ($\sigma_{blur} = 0.6$).

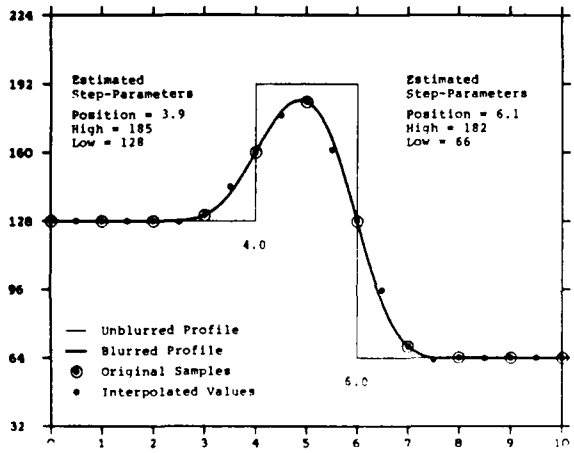


Fig. 5-b. Detection and estimation of step-edges spaced 2 pixels apart ($\sigma_{blur} = 0.6$).

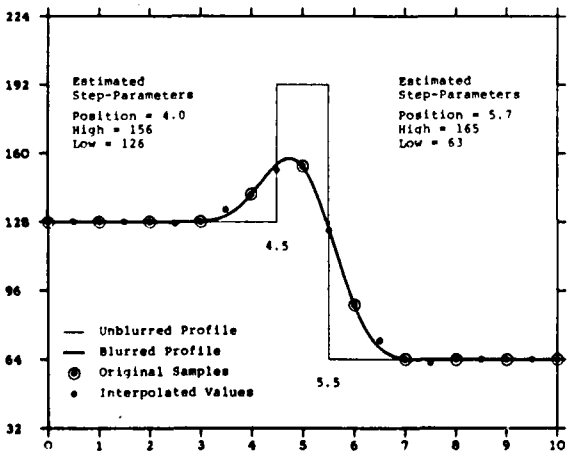


Fig. 5-c. Detection and estimation of step-edges spaced 1 pixel apart ($\sigma_{blur} = 0.6$).

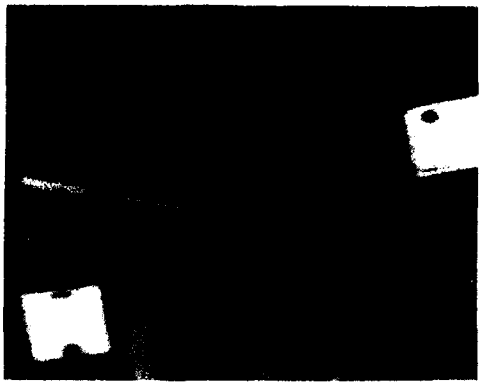


Fig. 6-a. Bin of Parts : Original Image (128 x 128)

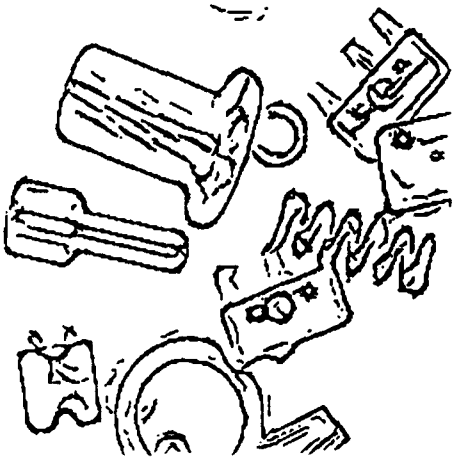


Fig. 6-b. Bin of Parts : Edge Image without Interpolation (adapted from [9])

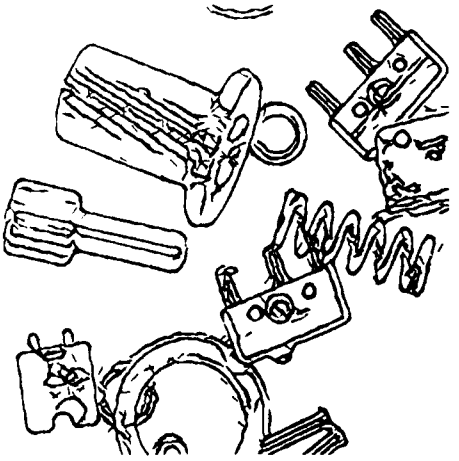


Fig. 6-c. Bin of Parts : Edge Image with Interpolation

Fig. 7-a. Indoor Scene : Original Image (256 x 256)

Fig. 7-b. Indoor Scene : Edge Image without Interpolation (adapted from [9])

pixel).

Fig. 7-a is the original (256 x 256) image of an indoor scene comprising of a cup with a pencil inside it, a telephone, and a book, all placed on a desk. Fig. 7-b is the corresponding edge-detector output adapted from [9]. Fig. 7-c is the result of applying the same edge-detector with identical parameters to the interpolated image. We would like to bring to the reader's attention the improved resolution in the central portion of the flower on the cup and in the fringes of the telephone dial. Notice that the push buttons on the telephone are less rounded in Fig. 7-c. Also notice the newly detected edge along the length of the pencil. Lastly, we point out that Fig. 7-c seems to have more false positives than Fig. 7-b. σ_{blur} for this image was estimated to vary between 0.5 pixel and 1.5 pixel.

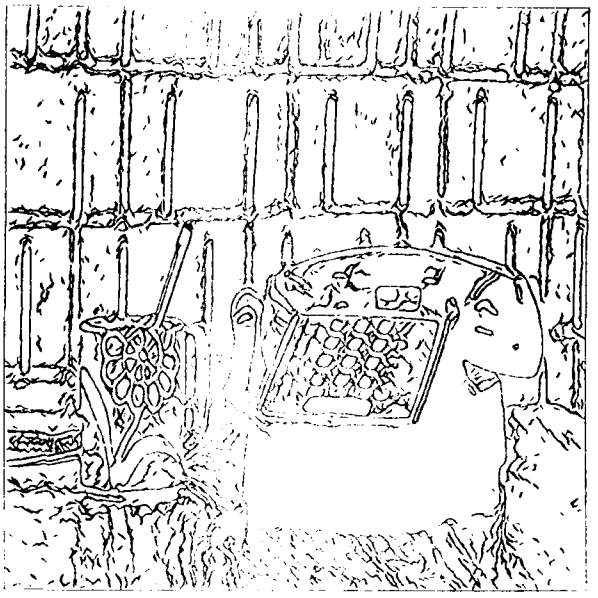


Fig. 7-c. Indoor Scene : Edge Image with Interpolation

We conclude [illegible] with an illustration of the algorithm's [illegible] on an image of printed text. Fig. 8-a is the original (192 x 256) image, Fig. 8-b is the corresponding edge image without prior interpolation and Fig. 8-c is the edge image with interpolation. The edge-detector outlined in [9] was employed in both cases. σ_{blur} for this image was estimated to be 0.8 pixel.

V. Discussion

We have [illegible] demonstrate a technique to improve the resolution capability of any edge-detector. It involves [illegible] reconstruction of the underlying intensity surface prior to step-edge detection. Although we carried out explicit interpolation in the examples presented, the interpolation step can often be incorporated in subsequent processing. This is particularly simple if the first step of the edge-detector involves the application of a linear space-invariant operator (such an operator can always be represented as a convolution mask). Of course, if the edge operator is already based on implicit interpolation, e.g. the derivative of an interpolation function, then the preprocessing may be redundant.

Historically, edges in images have been classified as step-edges, roof-edges and line-edges (see [1]) and it has been implicit in every discourse on the subject, to the best of our knowledge, that each of these categories requires a special purpose detector. We have demonstrated that a line-edge is simply a pair of interacting step-edges which can be individually detected. However, the interaction between step-edges causes their parameter estimates to be biased ---

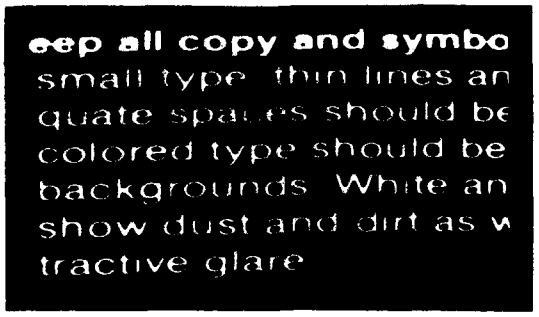


Fig. 8-a. Text : Original Image (192 x 256)

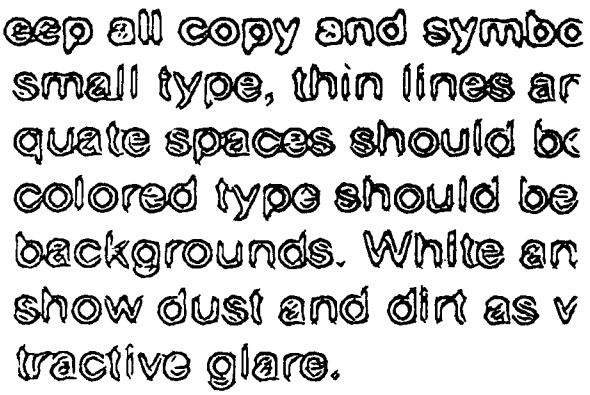


Fig. 8-b. Text : Edge Image without Interpolation

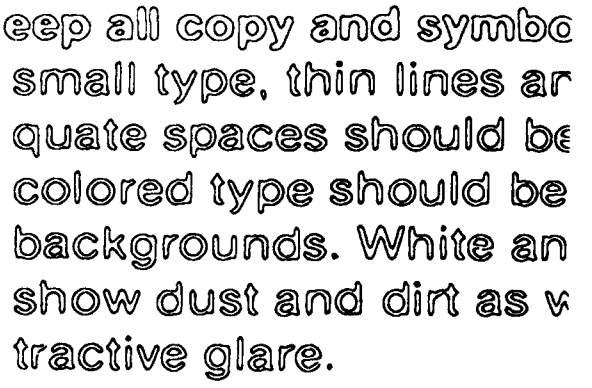


Fig. 8-c. Text : Edge Image with Interpolation

the more the interaction, the larger are the biases. First order corrections for these biases can be subsequently obtained from knowledge of the effective blurring function associated with the imaging system.

The discrimination of a step-edge from noise in an image depends on the magnitude of the blurred step. While this magnitude is the same as that of the corresponding unblurred step when the edge is isolated, it may be substantially lower when the step-edge is interacting, as is evident from Fig. 5. The contribution of the line-component $\{f(x) = 0, h, 0 \quad x<-w/2, -w/2\leq x\leq w/2, w/2<x\}$ to the height of the blurred step is $\int_{-w/2}^{w/2} h \frac{1}{\sqrt{2\pi\sigma_{blur}^2}} \exp(\frac{-x^2}{2\sigma_{blur}^2})\, dx$, assuming a normalized Gaussian blurring function.

For a largely bandlimited image, resolution is determined by the degree of blur. There exists a large body of work on edge-detection which recommends implicit or explicit smoothing of the image samples as the first step. The most commonly used smoothing function is the 2-D Gaussian (see [11]). It can easily be shown that smoothing with this function would result in the smoothed image having an effective Gaussian blurring function with a variance equal to the sum of the variance of the original Gaussian blurring function and that of the Gaussian smoothing function. Hence, resolution between interacting step-edges, and their detection will both suffer. Also, the biases in their position and intensity estimates will increase. The extent of this deterioration will of course depend on the magnitude of σ_{smooth} (σ_{smooth} typically has a suggested lower bound of 1.0 (see [5])).

Although the proposed approach to detect line-edges is more general than any special purpose line-detector, we should also expect it to be more sensitive to noise with respect to false positives, particularly at high-gradient smoothly-shaded regions, false negatives, and the parameters of the true positives. As is evident from Fig. 3, improvement in resolution is necessarily accompanied by a shrinking of the support used on the interpolated profile for detection. This results in reduced robustness to local noise-induced fluctuations in the reconstructed intensity profile. Thus, we have a trade-off between robustness and generality. This should come as no surprise considering that matched-filtering, wherein noisy patterns are categorized based on their closest "match" to noiseless representatives of the different classes, is well known to be optimal (see [6]).

We conclude by noting that if the suggested interpolation cannot be conveniently incorporated into subsequent processing, then it is computationally desirable to exploit the separability of the interpolation filter, i.e. a 2-D separable filter can be implemented as two successive 1-D filters in the x and y directions. It is also advisable to avoid redundant computations like multiplication by a 1 or a 0, or the addition of a 0. To get an idea of the resulting savings, consider the filter shown in Fig. 4. A naïve application of this filter would require approximately 100 operations per interpolated value, while a more sophisticated implementation would require only 5. This is easily verified by noting that every interpolated value can be obtained by the application of the 1-D filter $\frac{1}{16}\{0\ -1\ 0\ 9\ 16\ 9\ 0\ -1\ 0\}$ to the partially interpolated image.

Appendix : Interpolation using 2-D Separable Filter

We show here that for any sampled surface of the form

$$I(x, y) = \sum_{i=0}^{3}\sum_{j=0}^{3} c_{ij} x^i y^j \sum_n\sum_m \delta(x-n)\,\delta(y-m)$$

$$\text{where } \delta(k) = \begin{cases} 1 & \text{if } k = 0 \\ 0 & \text{if } k \neq 0 \end{cases}$$

we will get perfect interpolation using the separable filter $r(x)\,r(y)$ where $r(x)$ is designed to reconstruct sampled 1-D signals in the space spanned by $\{1\ x\ x^2\ x^3\}$. The reconstructed signal can be expressed as

$$\begin{aligned} R(x, y) &= \int\int I(\alpha, \beta)\, r(x-\alpha)\, r(y-\beta)\, d\alpha\, d\beta \\ &= \int\int \left\{ \sum_{i=0}^{3}\sum_{j=0}^{3} c_{ij}\alpha^i\beta^j \sum_n\sum_m \delta(\alpha-n)\,\delta(\beta-m) \right\} \\ &\qquad r(x-\alpha)\, r(y-\beta)\, d\alpha\, d\beta \\ &= \sum_{i=0}^{3}\sum_{j=0}^{3} c_{ij} \left[\int \alpha^i\, r(x-\alpha) \sum_n \delta(\alpha-n)\, d\alpha\right] \\ &\qquad \left[\int \beta^j\, r(y-\beta) \sum_m \delta(\beta-m)\, d\beta\right] \\ &= \sum_{i=0}^{3}\sum_{j=0}^{3} c_{ij} \left[\sum_n n^i\, r(x-n)\right] \left[\sum_m m^j\, r(y-m)\right] \\ &= \sum_{i=0}^{3}\sum_{j=0}^{3} c_{ij}\, x^i\, y^j \quad \text{as } [.] = x^i,\ y^j \text{ by design} \end{aligned}$$

Acknowledgement

The author would like to thank his advisor, Tom Binford, for his help. This work was prompted by discussions with Gerard Medioni and Jean Ponce.

References

[1] K.E.Atkinson, *An Introduction to Numerical Analysis*. John Wiley & Sons, New York, 1978.

[2] R.N.Bracewell, *The Fourier Transform and its Applications*. McGraw-Hill, New York, 1978.

[3] H.B.Barlow, "Reconstructing the Visual Image in Space and Time," *Nature*, Vol.279, May 1979, 189-190.

[4] M.Brady, "Computational Approaches to Image Understanding," *Computing Surveys*, Vol.14, No.1, March 1982, 3-71.

[5] F.J.Canny, "Finding Edges and Lines in Images" AI-TR 720, M.I.T. A.I. Lab., June 1983.

[6] J.W.Goodman, *Introduction to Fourier Optics*. McGraw-Hill, New York, 1968.

[7] R.M.Haralick, "Digital Step Edges from Zero Crossing of Second Directional Derivatives," *IEEE Trans. PAMI-6*, No.1, Jan. 1984, 58-68.

[8] M.H.Hueckel, "A Local Visual Operator which Recognizes Edges and Lines," *J. ACM*, Vol.20, No.4, Oct. 1973, 634-647.

[9] V.S.Nalwa and T.O.Binford, "On Detecting Edges," *IEEE Trans. PAMI-8*, No.6, Nov. 1986, 699-714.

[10] W.K.Pratt, *Digital Image Processing*. John Wiley & Sons, New York, 1978.

[11] V.Torre and T.Poggio, "On Edge Detection," *IEEE Trans. PAMI-8*, No.2, March 1986, 147-163.

Detection, Localization and Estimation of Edges[1]

J.S. Chen and G. Medioni

Institute for Robotics and Intelligent Systems
Departments of Electrical Engineering and Computer Science
University of Southern California
Los Angeles, California 90089-0273

Abstract

We present a method to detect, locate and estimate edges in a one dimensional signal. It is inherently more accurate than all previous schemes as it explicitly models and corrects interaction between nearby edges. The method is iterative with initial estimation of edges provided by the zero crossings of the signal convolved with a **LoG** (Laplacian of Gaussian) filter. The necessary computations necessitate knowledge of this convolved output only in a neighborhood around each zero crossing and in most cases, could be performed locally by independent parallel processors. Even though the model used for edges is very crude, the results are very impressive and allow us to reconstruct the signal extremely well. We show results on 1-dimensional slices extracted from real images, and on images which have been processed independently in the row and column directions. We provide an analysis of the method including issues of complexity and convergence, and outline directions of future research.

1 Introduction

In order to achieve image understanding, biological or computer system must relate the raw input data (viewer-centered) to the physical events that cause it, in particular the objects being observed (object-centered). The first step in this complex sequence of operations is to obtain a compact description of the input image. At this early stage of processing, it is essential for the resulting representation to be *complete*, that is to capture all of the information contained in the original image. It should be therefore possible and easy to *reconstruct* the original signal from the representation, as suggested by [Marr79,Marr80]. The other necessary characteristic of the representation is that it should be *meaningful*, that is to make explicit the important properties of the imaged scene. As noted by many authors [Binford81,Brady82,Marr80], physical edges of the objects are fundamental descriptions of these objects as they relate to transitions in surface orientation or texture. These edges are very likely to be projected onto edges in the intensity data received by the sensor. From the short analysis above, it therefore appears that the initial task of a vision system is the *detection, localization* and *characterization* of the local 2-dimensional edges in intensity.

We begin by reviewing the major approaches taken by previous researchers to accomplish edge detection, localization and estimation. We then point out that, in spite of the claims of optimality, the resulting edges are disappointing because the model used is unrealistic, namely that of a single isolated edge. We then proceed to introduce a model formulation which explicitly takes into acount the interaction between edges, and develop an iterative scheme to correct this interaction. After that, we provide an analysis of the implementation, addressing issues of complexity, stability, and convergence. The next section illustrates our method on a few examples both in 1-D and 2-D. For each image, we also provide a reconstructed picture from our edge description. Finally, we provide some conclusions and outline further research directions.

2 Previous Work

From the very early days of computer vision, edge detection was recognized as necessary, with a simple implementation of a gradient function [Roberts65]. Noise sensitivity forced the inclusion of a smoothing step before differentiation [Rosenfeld71]. A more rigorous approach based on the ideal model of an edge led to the Hueckel operator with the first claim of optimality. More claims of optimality have been made by Marr and Hildreth [Marr80] , extending the initial work of Marr and Poggio [Marr79] and the edges from this detection scheme have become a standard gauge against which all other methods are compared. The basic approach is to convolve the signal with a rotationally symmetric Laplacian of Gaussian mask (sometimes approximated by a Difference of Gaussians), and to locate zero-crossings of the convolution. A recent paper by Torre and Poggio [Torre86] judiciously points out that better results may be obtained by using 2 directional filters with directional derivatives, especially in the neighborhood of corners.

[1]This research was supported by the Defense Advanced Research Projects Agency under contract number F33615-84-K-1404, monitored by the Air Force Wright Aeronautical Laboratories, Darpa Order No. 3119.

Other different approaches have made claim of optimality or shown results subjectively comparable or superior to the Marr-Hildreth zero-crossing detector. Among them :

1. Dickey and Shuningham [Dickey77] define an edge as a step discontinuity between regions of uniform intensity and show that the ideal filter is given by a prolate spheroidal wave function. Lunscher and Beddoes [Lunscher86] show that the Marr-Hildreth filter is nearly identical, but simpler to study and implement.
2. Haralick [Haralick80,Haralick81,Haralick82] locates edges at zero crossings of the second directional derivative in the direction of the gradient. Derivatives are computed by interpolation of the sampled intensity values. On some images, the resulting edges are visually better than the ones from the Marr-Hildreth detector. Torre and Poggio [Torre86] give an excellent analysis of the difference between the Laplacian and the second derivative in the direction of the gradient.
3. Canny [Canny83] shows that, in 1-D, the optimal filter, according to his criterion, is a linear combination of four exponentials, well approximated by a first derivative of a Gaussian. In 2-D images, he proposes to use a combination of such filters with varying length, width and orientation. Since it is a first derivative operator, it requires further thinning and thresholding. (Our implementation of it uses modules from the "Linear" feature extraction program [Nevatia80].) The results on real images are quite good. This can be attributed, in our view, to the fact that the thinnest mask possible is used, therefore minimizing edge interaction effects.
4. Shen and Castan [Shen86] recently proposed an "optimal" linear filter, in which images are convolved with the smoothing function $f(x) = -\frac{1}{2}ln(b)b^{|x|}$ prior to differentiation. They claim better localization than Marr-Hildreth zero-crossing detector. On real images, good edges are obtained after a few extra filtering steps tuned by the user.

It is quite important to note that, in spite of these claims of optimality, edges are quite disappointing. Also, precisely because these edges are poor, very few authors attempt to reconstruct the original signal.

It is our belief that edge detection should not crucially depend on small differences in the smoothing function, and also that trying to design a filter that will always respond with a zero-crossing at the correct location is as unrealistic as trying to accomplish image understanding in one step. The main problem with all previous approaches is that they assume an unreasonable model, namely an *isolated* edge, when we all observe that edges often occur close together.

The following sections explain and illustrate how to obtain good edges from an initial estimate by explicitly modelling edge interaction.

3 Description of the Method

3.1 Informal Description

Our purpose is to accurately estimate both the location and magnitude of edges, initially in a one dimensional signal. The two sources of error in accurately locating edges is interaction between them, and the lack of symmetry of the edge profile, and it is our observation that edge interaction is the larger of the two. Effects of edge profiles on localization are studied in another paper [Medioni86]. To correct this interaction, we therefore use a very simple model in which edges are perfect steps. Each step is completely specified by its location and signed magnitude. As a result, the convolution of the signal with a **LoG** filter can be expressed at any point by a weighted sum of the responses from each discontinuity.

During the initialization phase, the number of discontinuities is given by the number of zero crossings, the location by the (interpolated) position of the zero crossings, and the magnitude by an estimation involving convolution of the signal with the first derivative of a Gaussian, as explained in detail in the next subsection.

The iteration proceeds by hypothesize and verify : Supposing that the signal is well approximated by the initial train of discontinuities, we can synthesize the corresponding convolution with a **LoG** mask. Due to the interaction between nearby edges, the correct location of a discontinuity is no longer at the zero crossing position but at some nearby location corresponding to a nonzero level crossing. Once the new locations are computed, we estimate the corresponding magnitude of the step. This iterative improvement continues until all values have converged to a stable position, or until the improvement become negligible.

It is important to notice that we only use the values of the signal convolved with the **LoG** filter in a small neighborhood around each zero crossing to perform our computation. It is also interesting to note that, even though our model is very crude, it estimates the edges so nicely that the reconstructed signal is a very acceptable approximation of the original, as shown in the results.

3.2 Mathematical Description

We use a simple model for edges, namely a step edge. Let the j^{th} step edge at location z_j with step size s_j is denoted by I_j :

$$I_j(x) = \begin{cases} c_j & x < z_j \\ c_j + s_j & x \geq z_j \end{cases} \quad (1)$$

with the additional constraint that $c_{j+1} = c_j + s_j$.

And we assume that the signal can be well approximated by the function $I(x)$ which is the sum of all edges :

$$I(x) = \sum_{j=1}^{n} I_j(x) \tag{2}$$

The second derivative of a Gaussian is :

$$L(x) = \frac{1}{\sigma^2}(1 - (\frac{x^2}{\sigma^2}))e^{-\frac{x^2}{2\sigma^2}} \tag{3}$$

Therefore the convolution of the approximating signal and of a **LoG** filter is :

$$F(x) = I(x) * L(x) = \frac{1}{\sigma^2}\sum_{j=1}^{n} s_j(x - z_j)e^{-\frac{(x-z_j)^2}{2\sigma^2}} \tag{4}$$

Each term in the sum above represents the convolution of a single isolated step edge with a **LoG** filter. It is exactly the first derivative of a Gaussian function, in which the zero crossing is located at z_j, the peak and valley are at a distance of σ on either side of z_j, and the amplitude is proportional to s_j, the size of the step. If adjacent edges are far apart, then the contribution at any point is due to a single edge, since the exponential terms corresponding to the other edges are extremely small.

In particular, $F(z_k) = 0$, and the position of each zero crossing accurately reflects the location of the edge. Most of the time, however, edges will interact, and the value at true edge location z_k is no longer 0, since

$$F(z_k) = \frac{1}{\sigma^2}\sum_{j=1}^{n} s_j(z_k - z_j)e^{-\frac{(z_k-z_j)^2}{2\sigma^2}} \tag{5}$$

The above equations naturally suggest an iterative scheme to obtain the true location of edges if we have a good initial estimate of them :

Given an estimate $I^{(t)}$ of our edges at iteration t, then we can find the new and more accurate location $z_k^{(t+1)}$ of the k^{th} edge as the point whose value is $F(z_k^{(t)})$.

In order to compute equation (5) above, we also need an estimate of step size s_j. It is tempting to use equation (4) again to perform this estimation. Our experiments with such an approach gave rise to unstable results due to the nature of the equations. Instead, this is accomplished using a convolution with the first derivative of a Gaussian.

The first derivative of a Gaussian is :

$$D(x) = -\frac{x}{\sigma^2}e^{-\frac{x^2}{2\sigma^2}} \tag{6}$$

The output of the convolution of our train of edges with this filter is :

$$H(x) = I(x) * D(x) = \sum_{j=1}^{n} s_j e^{-\frac{(x-z_j)^2}{2\sigma^2}} \tag{7}$$

By evaluating this expression at n points, such as the detected edge points, we obtain a linear system of equations :

$$H(z_k) = \sum_{j=1}^{n} s_j e^{-\frac{(z_k-z_j)^2}{2\sigma^2}} \qquad k = 1, 2, ..., n \tag{8}$$

In matrix notation, we have :

$$\mathbf{H} = \mathbf{M} \times \mathbf{S} \tag{9}$$

where,

$\mathbf{H}$ is a column vector where $H_k = H(z_k)$,
$\mathbf{S}$ is a column vector where $S_j = s_j$, and
$\mathbf{M}$ is a $n \times n$ matrix with $M_{ij} = e^{-\frac{(z_i-z_j)^2}{2\sigma^2}}$

Note that $\mathbf{M}$ is a extremely well behaved matrix, symmetric, positive definite, with 1's on the diagonal. For all pratical purposes, it can also be considered as a band matrix. The inversion of this matrix is quite straightforward, leading to a set of values for the step sizes.

We can now give the iteration scheme in detail :

1. *Initial estimation*
   At iteration 0, the locations of the edges are given by the zero crossings of the input signal convolved with a **LoG** filter.
2. *Iteration formula*
   At any iteration t, $\mathbf{S}$ is obtained by computing $\mathbf{M}^{-1} \times \mathbf{H}$. This provides an estimate for the step sizes. Then $z_k^{(t+1)}$ is the point closest to $z_k^{(t)}$ such that

$$F(z_k{}^{(t+1)}) = \sum_{j=1}^{n} s_j^{(t)}(z_k^{(t)} - z_j^{(t)})e^{-\frac{(z_k^{(t)}-z_j^{(t)})^2}{2\sigma^2}} \tag{10}$$

3. *Stopping criterion*
   Since the behavior of the iteration scheme is quite stable and most of the interaction of edges can be corrected in the first few steps, the iteration can be stopped either simply after a fixed number of iteration, or with some simple error criterion such as comparing the difference between the actual convolution output with the modelled result at a few points such as those level crossing points.

4 Detailed Analysis

4.1 Complexity of the Algorithm

The algorithm consists of two parts. The first part is the initial detection of the zero crossings of the convolution of the signal with the **LoG** mask. Many fast algorithms have been proposed for the convolution with the **LoG** filters, see [Chen86]. The zero-crossing detection, usually in

subpixel precision, can be done by various interpolation schemes, and there is always a trade-off between accuracy and efficiency. We have tried several interpolation methods (cubic, quadratic, and linear), and feel that the improvements due to higher order terms do not warrant the extra computation time, so we adopted a linear interpolation.

The second part is the iteration scheme. The computation includes :

1. estimation of step sizes by solving matrix equation (9),
2. calculation of the new levels using equation (5),
3. finding the new level-crossings in the neighborhood of corresponding previous level crossings.

Calculation of a new level for a particular edge from equation (5) requires the summation of the effect from all the edges, but for practical consideration, we only have to include the nearby edges since the rate of exponential decrease is very fast. Finding the new level crossing position can be done efficiently either by searching along the curve in the neighborhood of a previous location, or by extrapolation. So the major concern should be on the estimation of step sizes.

Since the complexity of solving linear system equations is of the order n^3 , where n is the number of equations, a first look of equation (9) seems to indicate that the complexity is of the order of n^3, where n is the number of edges. But the matrix M is a well behaved matrix. It is symmetric , positive definite, and, for all practical purpose, since the decrease along both sides of the diagonal is exponential, it can be approximated by a band matrix. Experimental results shows that the full matrix can be very well approximated by a band matrix with bandwidth of 5, which says that no more than 3 edges interact at any given point. So the complexity of solving equation (9), by using LU-decomposition, is only of the order of n, the number of edges, instead of n^3.

It is also worth mentioning that if one of the off-diagonal elements is very small, then the M matrix can be broken at that point into 2 submatrices which can be inverted separately. We are also investigating non-exact methods to invert such a matrix that would run in $O(1)$, i.e. constant time, on a parallel machine.

4.2 Stability and Convergence

Experimentally, the algorithm is quite stable, and most of the edge interaction effect can be corrected in the first few iterations. Sometimes, however, the iteration for some edges, usually only one or two, does not converge and oscillate around the correct edge position. The amplitude of oscillation is small but not entirely negligible. However, this oscillation can be easily detected and the correct edge position can be obtained by averaging values once oscillation starts.

4.3 Limitations of the Method

The first and basic limitation of our method is the spatial frequency which can be resolved: As the **LoG** is a band pass filter, it can only see edges in its bandwidth. This is illustrated in the example of a "bump", showing the width estimation of a bump edge as σ increases: as long as σ is small enough, the width is correctly computed, and after that, grows with σ.

Since differentiation is typically an ill-posed problem [Torre86], regularization is necessary, in a filtering step. Consider the example of a bump of limited width, say 1 sampling spacing, then for different filters, such as different σ for a **LoG** filter, it has different interpretations. In terms of the limitation of our method, as the size of the **LoG** filter increases, i.e. σ increases, we lose the high frequency contents of the bump and give a different interpretation which is another bump with larger width smaller height.

The other limitation also comes from the issue of the filter size, as the size increase, the smoothing effect increases, therefore some edges will not be detected at the initial zero-crossing detection. Our method only tries to correct those initially detected edges, and we are trying to interpret the signal at a particular level where the resolution is decided by the value of σ.

Since we start with a very simple model for an edge, namely an ideal step edge, there are limitations when the model does not fit. For example, in the case where a "sloped" edge is involved such as a ramp, or when there is a slope difference on the two sides of a step, our model can no longer fit and just try to make a best interpretation with the ideal step edge model. Our experimental results shows that in the case of a symmetric ramp, we will get an edge in the middle of the ramp, and the step size is underestimated since the intensity change of a ramp is averaged over the interval of the ramp. We believe that these other types of edges can be detected and correctly estimated, as presented in [Medioni86].

5 Results

In this section, we show some comparative results from pure zero-crossing detection of **LoG** convolution with the input signal, and from our algorithm. We first show results on some ideal signals, then we show some results on some 1-dimensional signal slices extracted from a real 2-dimensonal image and we also produce the reconstructed signals. Finally, we demonstrate the behavior of the method in the presence of noise.

5.1 Ideal Signals

We first illustrate our method on a ideal "bump", which consists of 2 discontinuities of equal magnitude but opposite sign, as shown on figure 1a. The width of the bump is 3 pixels, its magnitude 100 units. Figure 1b shows the location of the zero crossings as a function of σ, the space constant of the LoG mask, and it exhibits the expected behavior that the 2 zero crossings "repel" each other more and more as σ increases. Figure 1c shows the effects of our correction algorithm for increasing values of σ. The detected width is constant up to a certain value, then increases with σ, due to the limits on the spatial frequency that each filter can resolve. In figures 1d and 1e, we show a plot of the difference between the convolved output of the "true" bump (width = 3, height = 100) and the estimated one for values ranging between $zx - 0.3$ and $zx + 0.3$ for the edge location, where zx is the true position of the edge, and between 70 and 130 for the height. For $\sigma = 1.4$, there appears to be a clear "pit" corresponding to the correct values of width and height of the bump. But for $\sigma = 6.0$, the pit has disappeared and is replaced by a gentle sloping ravine line showing an interpretation of the signal as a lower and wider bump. Note it is not easy to relate directly these error diagrams to our iterative scheme.

We now show the other possible 2-edges pattern which we call a "staircase". It is a succession of 2 edges of the same height and sign separated by a given width shown in figure 2a. In some range of σ, this pattern generates 3 zero crossings, the middle one corresponding to an inflection point rather than a real edge. We presented a method to detect such occurences in a previous paper [Medioni86] : for inflection points, $f' \times f''' > 0$, where f' and f''' are the input signal convolved with the first and the third derivative of a Gaussian respectively. It happens, however, that some valid edges are discarded by this method when σ is so large that interaction in f' produces the wrong sign. For that reason, here, we prefer not to use this test and consider all zero crossings as potential edges.

In figure 2b we show the initial zero-crossing detection with respect to different level of σ , we can observe 3 zerocrossings between $\sigma = 2.0$ and $\sigma = 7.7$, the one in the middle is exactly the inflection point, i.e. a zero-crossing which does not correspond to an edge. Also note in figure 2b that before it goes out of the bound of resolving two edges, the edges start to interact and the zero-crossing locations become closer to each other and finally becomes only one single edge. We can see on figure 2c which shows the result of our method and the cancellation of the edge interaction effect. Note that the inflection point is quite "unstable", actually the step size of that inflection point is becoming relatively small, and we can remove it by simple thresholding. In figure 2d we apply a threhold value of $\frac{1}{20}$ of one of step sizes of the staircase, and we get a clean and accurate locations of the two edges before it goes beyond resolution ($\sigma = 7.7$).

5.2 Real Images

Figure 3a shows the original "cube" image of size 375 × 380 × 8*bits*. Figure 3b shows the zero-crossings of the image convolved with a rotationally symmetric LoG filter with $\sigma = 4.0$. In the last 3 figures, figures 3c, 3d, and 3e, the results are derived from first processing the image row-by-row, which is now a 1-dimensional processing and we can apply our algorithm directly, then processing the image column-by-column, finally combining the results from both directions. In particular, the reconstructed image figure 3e is derived by averaging the row-by-row and column-by-column reconstructed images. Figure 3c shows the combination of the zero-crossings of the original image convolved with 1-D LoG filter with $\sigma = 4.0$ in the horizontal and vertical directions. Applying our algorithm to refine this edge information, we get the result of figure 3d. Note that there some significant differences between figure 3c and figure 3d: In figure 3d, since our algorithm gets more accurate edge locations, the edges derived from two directional processes, namely row-by-row and column-by-column, almost coincide each other perfectly except at the bottom right of the cube, because these edges are pronounced ramp edges which our model can not fit.

Figure 4a and figure 5a are two slices of signals extracted from the "cube" image shown in figure 3a. Figure 4a is taken on the horizontal direction, figure 5a on the vertical. The reconstructed signals shown in figure 4b and figure 5b respectively are based on the initial zero-crossings of the original signals convolved with a LoG filter, and the step sizes from the original signals convolved with the first derivative of a Gaussian mask at the corresponding zero-crossing positions. With a very large σ of 6.0, we can see that the strong interaction of edges makes not only the zero-crossing positions be displaced from the exact edge locations, but also the step sizes be distorted. The reconstructed signals after our iteration scheme are shown in figure 4c and figure 5c respectively. The edges are those discontinuities (or "jumps") of the reconstructed signals and we can also see the step size associated with each edge. We also show in figure 4d and figure 5d respectively the edge locations as the iteration progresses. We can see not only that the iteration process converges, but also that most of the edge interaction is corrected in the first few iterations.

5.3 Ideal edges in noise

In figure 6a, we show a "bump image" of size 200 × 200 × 8*bits* added with Gaussian noise. The bump width is 5 pixels and the bump height is 100 units. The variance of the Gaussian noise is 20 units. In this example, we only process the image in row by row, and each row of the image is processed independently by a 1D process. By using σ of 7.0, the reconstructed image shown in figure 6b is obtained without iteration. Since we are applying a large σ,

ıe reconstruted bump width is much wider than the orig-
al one. In figure 6c, we show the reconstructed image
›tained by our method. We can see the reconstructed
ımp width is very consistent to the original bump width
; a result of accurate edge detection.
ı figures 7a and 8a, we show two horizontal slices ex-
acted from figure 6a. The reconstructed signals without
eration are shown in figures 7b and 8b respectively. And
ıe reconstructed signals by our method are shown in fig-
res 7c and 8c respectively.

6 Conclusions and Future Research

In this paper we have presented a technique to refine
lge positions after the detection of the zero-crossings of
ıe original signal convolved with a LoG filter. We also
stimate the magnitude and the sign of the step sizes mod-
.ling the edges which enables us to approximate and re-
›nstruct the original signal. The spectacular results show
ıat our algorithm works very well not only on some ideal
gnals, but also on real image data. We have shown the
fficiency of our algorithm and outlined the feasible par-
llelism. We have also shown the well-behavior of our
ıethod from its stability and convergent properties.

In section 4.3 we discussed the limitations of our method,
ıcluding the issue of the over-simplified model and the is-
ue of resolution. Even though we get very good results
sing a very simple edge model, namely the ideal step edge
ıodel, we believe better results can be obtain through the
se of a more general model. We also think that there exist
ıster and more efficent ways to solve the system of equa-
ions and to improve the convergence rate. To combine the
ıformation from different scales, i.e. using different σ's,
; another feasible solution for compensating the defect of
ur model, and furthermore, to resolve the problem of res-
lution.

Most important of all, to make our method really use-
ul, we must generalize the algorithm to 2-dimension. On
he subject of multiple scales processing, we believe that
›ur method provides a simple mechanism to established
orrespondences between different scales, as the position
›f edges are not affected by the values of σ, unless there is
; change of interpretation.

We have shown, in the previous section, good results
›f the processing of an image by using 2 one-dimensional
ilters, and this might be a feasible approach. We also
ntend to establish the correct formulation in 2-D, even
hough the solution might involve more complex methods.

References

[Binford81] Binford, T.O.,"Inferring Surfaces from Images", *Artificial Intelligence*, Vol 17, 1981, pp. 205-244.

[Brezins84] Brezins, V.,"Accuracy of Laplacian Edge Operators", *Computer Vision, Graphics and Image Processing*, Vol. 27, August 1984, pp. 195-210.

[Brady82] Brady, J.M., "Computational Approaches to Image Understanding", *Computing Surveys*, Vol. 14, 1982, pp 3-71.

[Canny83] Canny, F.R.,"Finding Edges and Lines in Images", *Tech. Report 720, MIT Artificial Intelligence Lab.*, June 1983.

[Chen86] Chen, J.S., Huertas, A., Medioni G., "Very Fast Convolution with Laplacian-of-Gaussian Masks", *Proceedings CVPR-86*, Miami Beach, Fl. 1986, pp 293-298.

[Davis75] Davis, L.S.,"Survey of Edge Detection Techniques", *Computer Graphics and Image Processing*, Vol. 4, April 1975, pp. 248-270.

[Dickey77] Dickey, F.M., Shanmugam, K.S., "Optimal Edge Detection Filter", *Appl. Opt.*, Vol. 16, No. 1, Jan. 1977, pp 145-148.

[Haralick80] Haralick, R.M., "Edge and Region Analysis for Digital Image Data", *Computer Graphics Image Processing*, Vol. 12, 1980, pp. 60-73.

[Haralick81] Haralick, R.M., "The Digital Edge", *Conf. Pattern Recognition Image Processing*, Dallas, TX. 1981, pp. 285-294.

[Haralick82] Haralick, R.M., "Zero-crossing of Second Directional Derivative Edge Operator", *SPIE Proc. Robot Vision*, Arlington, VA. 1982.

[Hildreth83] Hildreth, H.,"The Detection of Intensity Changes by Computer and Biological Vision Systems", *Computer Vision, Graphics and Image Processing*, Vol. 22, April 1983, pp. 1-27.

[Lunscher86] Lunscher, W.H.H.J., Beddoes, M.P., "Optimal Edge Detector Design I & II", *IEEE Tran. on Pattern Analysis and Machine Intelligence*, Vol. PAMI-8, No. 2, March 1986, pp 164-187.

[Marr79] Marr, D. and Poggio, T., "A Computational Theory of Human Stereo Vision", *Proceedings of the Royal Society of London*, B204, 1979, pp. 301-328.

[Marr80] Marr, D. and Hildreth, H., "Theory of Edge Detection", *Proceedings of the Royal Society of London*, B207, 1980, pp. 187-217.

[Medioni86] Medioni, G., Ulupinar, F., "Edge Detection Using Zero Crossings of Laplacian of Gaussian Bias Correction", To Be Published.

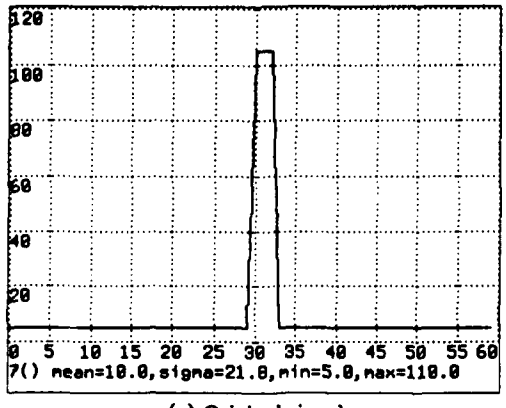
[Nevatia80] Nevatia, R., Babu, K.R., "Linear Feature Extraction and Description", *Computer Graphics and Image Processing*, Vol. 13, 1980, pp 257-269.

[Roberts65] Roberts, L.G., "Machine Perception of Three Dimensional Solids", *Optical nad Electro-Optical Information Processing*, MIT Press, 1965, pp 159-197.

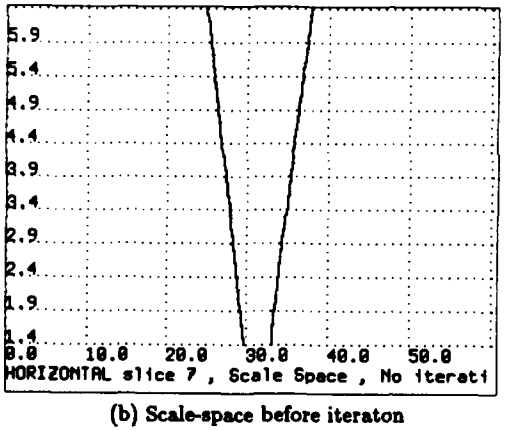
[Rosenfeld71] Rosenfeld, A., Thurston, M., "Edge and Curve Detection for Visual Scene Analysis", *IEEE Tran. on Computers*, Vol. C-20, 1971, pp 562-569.

[Shen86] Shen, J., Castan S., "An Optimal Linear Operator for Edge Detection", *Proceedings CVPR-86*, Miami Beach, Fl. 1986, pp 109-114.

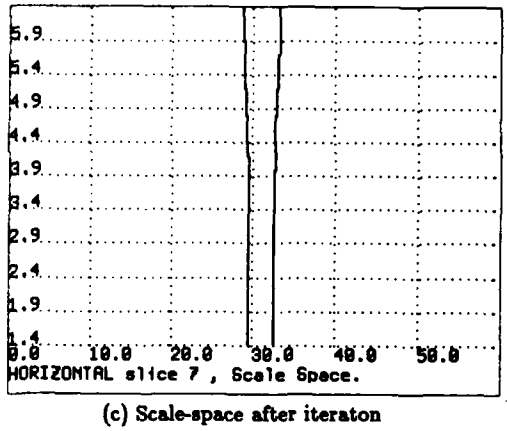
[Torre86] Torre, V., Poggio, T.A., "On Edge Detection", *IEEE Tran. on Pattern Analysis and Machine Intelligence*, Vol. PAMI-8, No. 2, March 1986, pp 147-163.



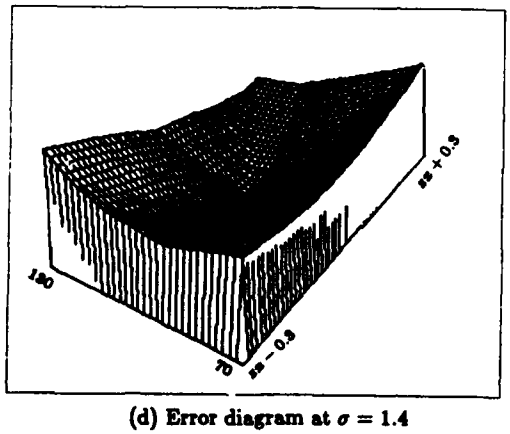
(a) Original signal



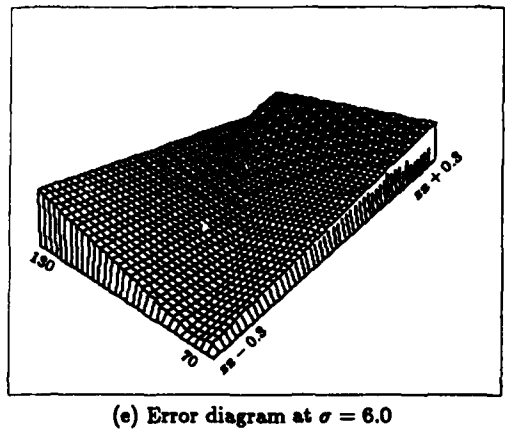
(b) Scale-space before iteraton



(c) Scale-space after iteraton

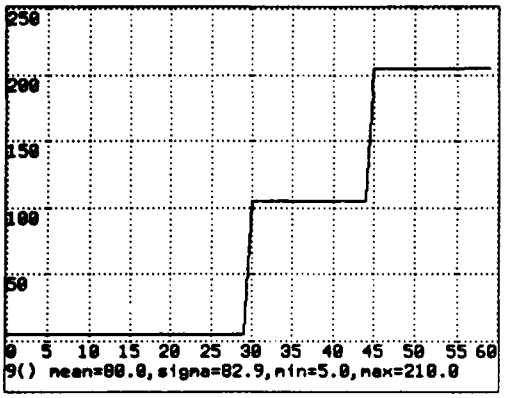


(d) Error diagram at $\sigma = 1.4$

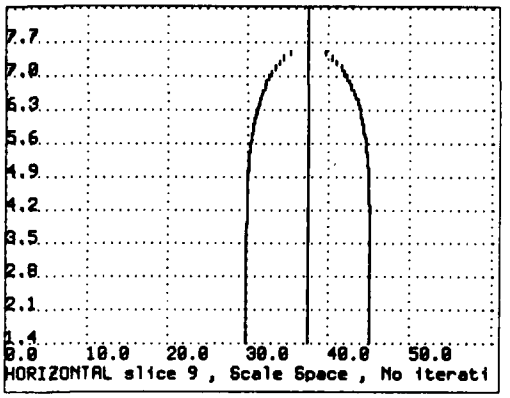


(e) Error diagram at $\sigma = 6.0$

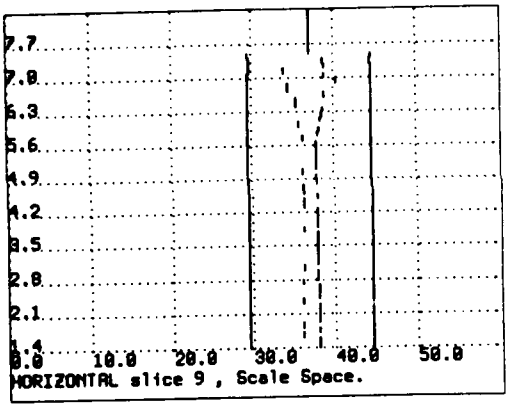
Figure 1: The "bump"



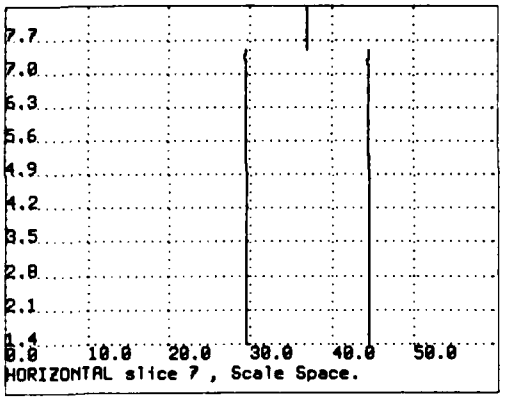
(a) Original signal



(b) Scale-space before iteraton

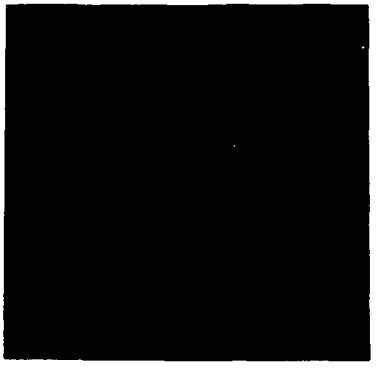


(c) Scale-space after iteraton without thresholding

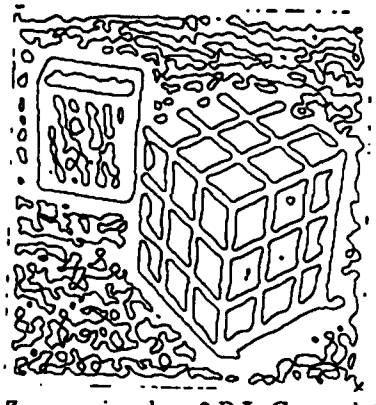


(d) Scale-space after iteraton with thresholding

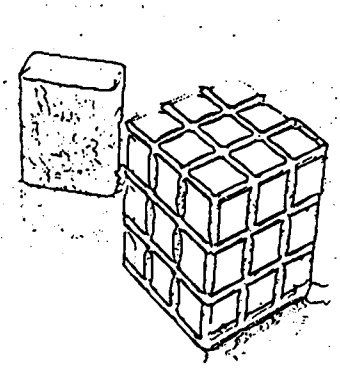
Figure 2: The "staircase"



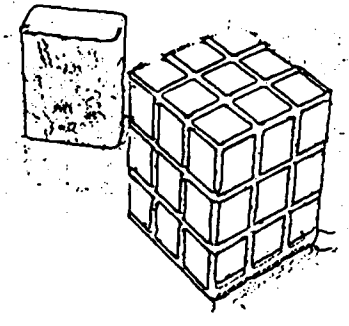
(a) Original image



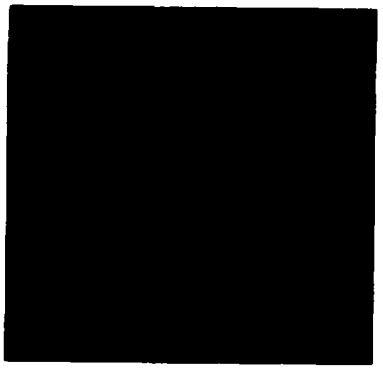
(b) Zero crossings by a 2-D LoG convolution



(c) Zero crossings by two 1-D detections

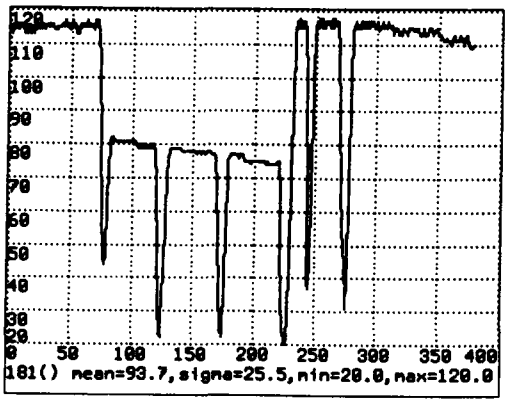


(d) Edges obtained from two 1-D processing of our method

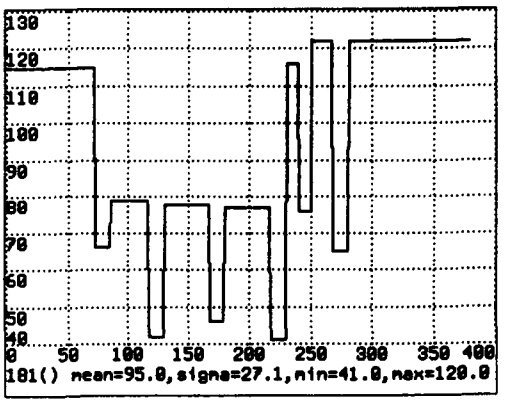


(e) Reconstructed image by our method

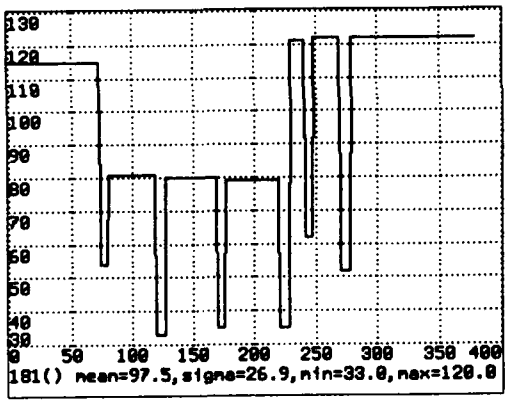
Figure 3: The "cube"



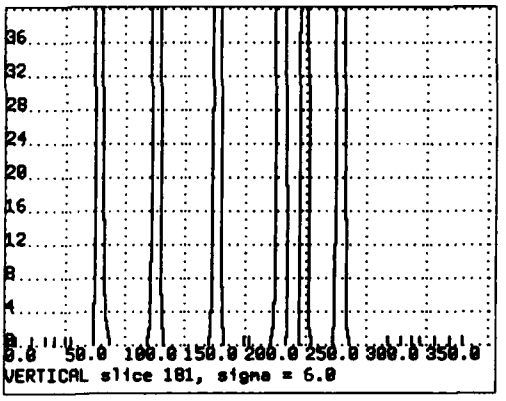
(a) Original signal



(b) Reconstructed signal before iteration

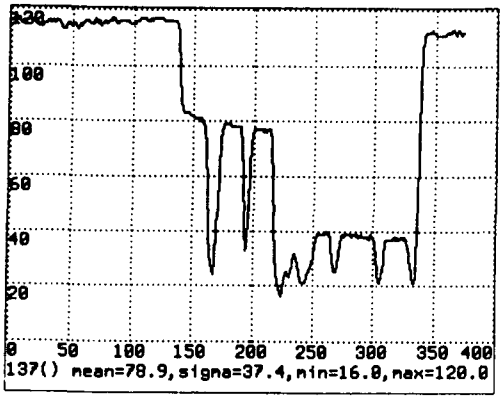


(c) Reconstructed signal by our method

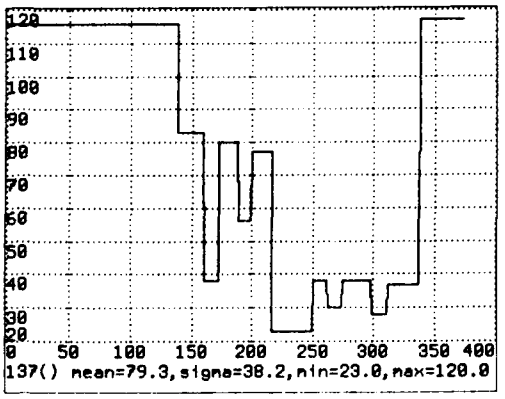


(d) Edge locations as iteration progresses

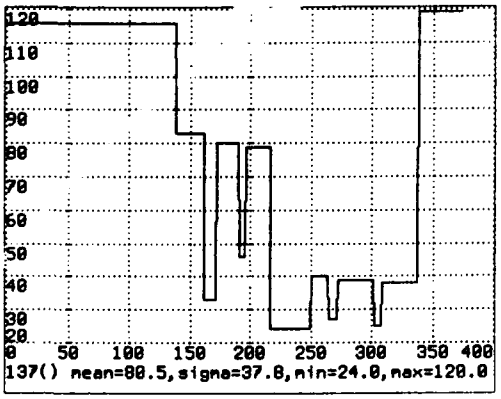
Figure 4: A vertical slice of the "cube" image



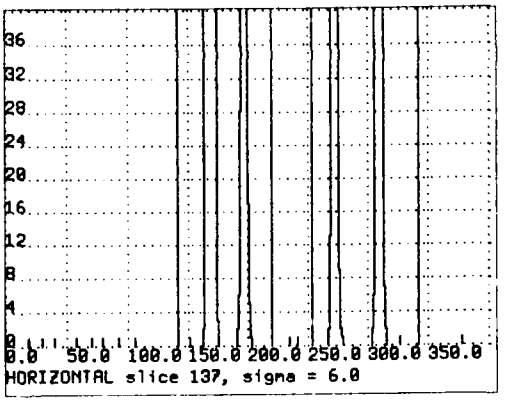
(a) Original signal



(b) Reconstructed signal before iteration

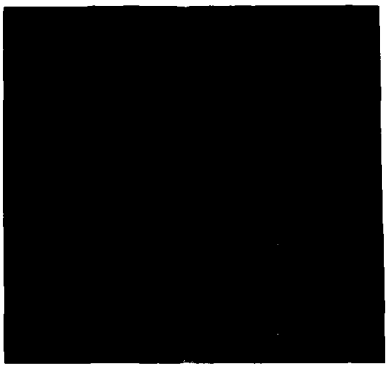


(c) Reconstructed signal by our method



(d) Edge locations as iteration progresses

Figure 5: A horizontal slice of the "cube" image

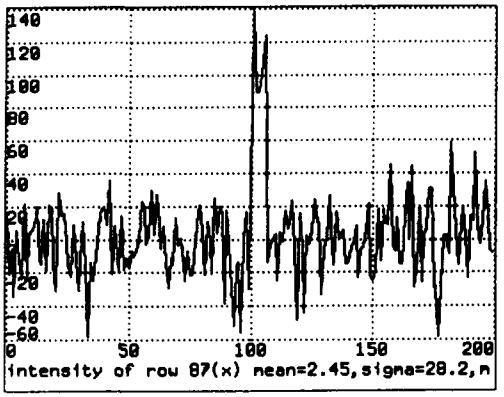


(a) Original image

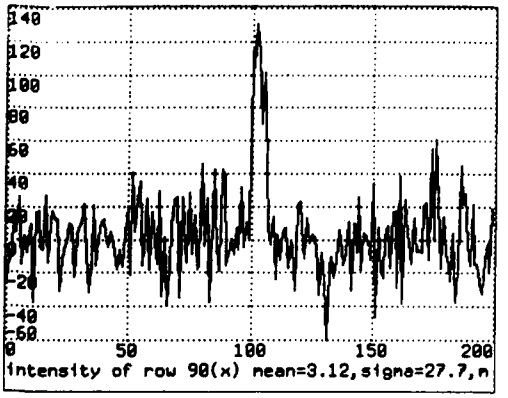
(b) Reconstructed image without iteration

(c) Reconstructed image by our method

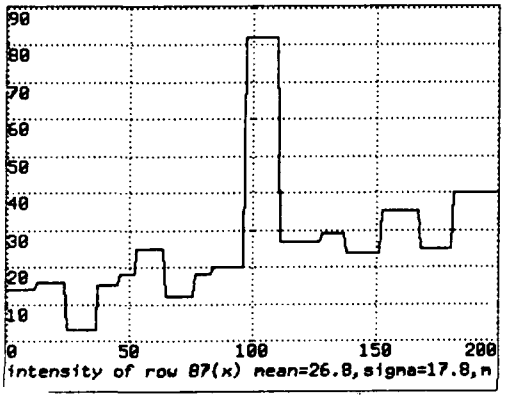
Figure 6: The "noisy bump"



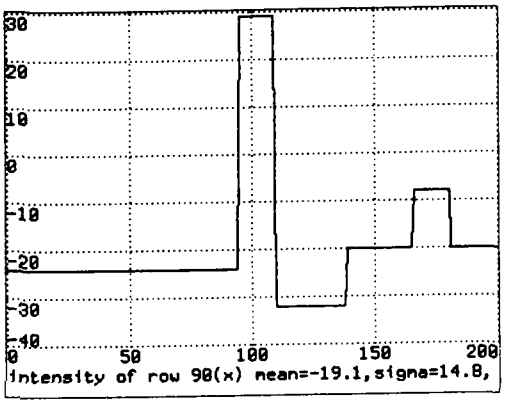
(a) A slice from figure 6a



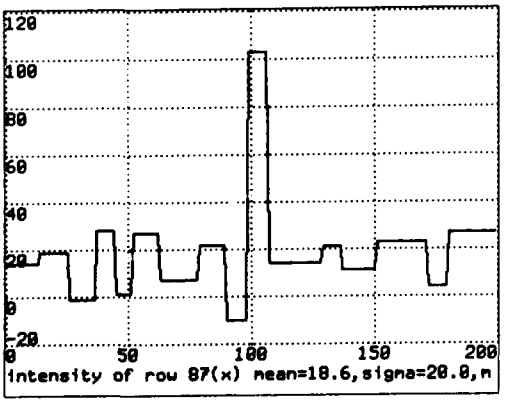
(a) Another slice from figure 6a



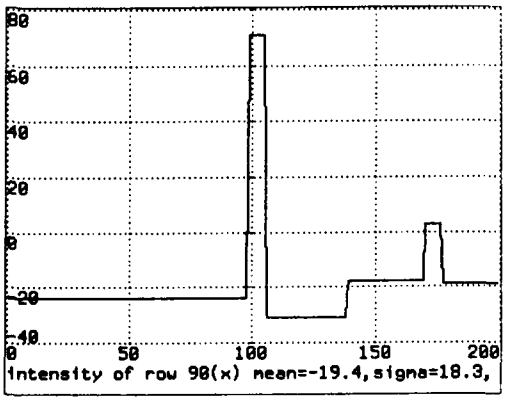
(b) Reconstructed signal without iteration



(b) Reconstructed signal without iteration



(c) Reconstructed signal by our method



(c) Reconstructed signal by our method

Figure 7: A slice of the "noisy bump"

Figure 8: Another slice of the "noisy bump"