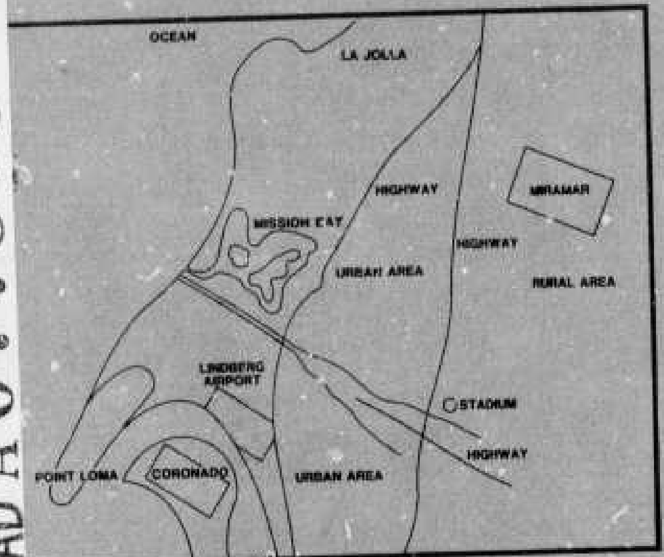
PROCEEDINGS: IMAGE UNDERSTANDING WORKSHOP

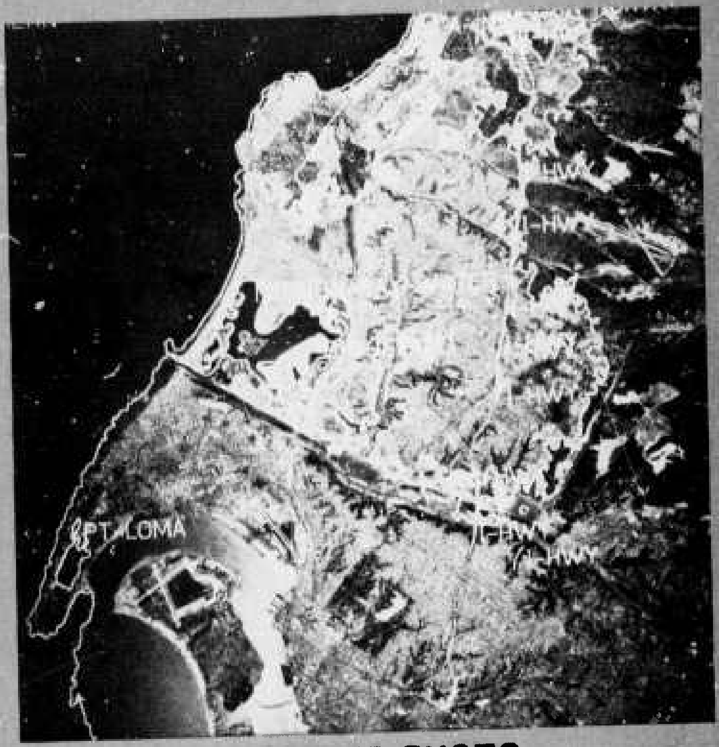
NOVEMBER 1979

Sponsored by:
Information Processing Techniques Office
Defense Advanced Research Projects Agency

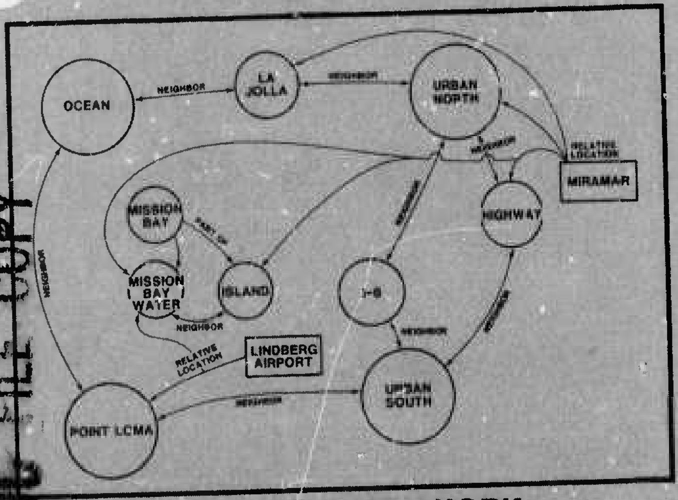
Hosted by:
University of Southern California
Image Processing Institute



SKETCH MAP



LABELED PHOTO



SEMANTIC NETWORK

Science Applications, Inc.

IMAGE UNDERSTANDING

Proceedings of a Workshop
Held at
Los Angeles, California
November 7-8, 1979

Sponsored by the
Defense Advanced Research Projects Agency

Science Applications, Inc.
Report Number SAI-80-974-WA
Lee S. Baumann
Workshop Organizer and
Proceedings Editor

This report was supported by
The Defense Advanced Research
Projects Agency under DARPA
Order No. 3454, Contract No. MDA903-78-C-0095
Monitored by the
Defense Supply Service, Washington, D.C.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. ~~REPORT NUMBER~~ (14) SAI-80-974-WA	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) (6) Proceedings of the Image Understanding Workshop, ~~November~~ 1979 Held at Los Angeles, California November 7-8 1979.		(9) 5. ~~TYPE OF REPORT & PERIOD COVERED~~ Semiannual Technical rept. Apr[redacted]1979 — Novem~~ber 1~~979
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) (10) Lee S. Baumann (Ed.)		8. CONTRACT OR GRANT NUMBER(s) (15) MDA903-78-C-0095, ARPA Order-3456
9. PERFORMING ORGANIZATION NAME AND ADDRESS Science Applications, Inc. 1911 North Fort Myer Drive, Suite 1200 Arlington, Virginia 22209		10. ~~PROGRAM ELEMENT, PROJECT, TASK~~ AREA & WORK UNIT NUMBERS ARPA Order No. 3456
11. CONTROLLING OFFICE NAME AND ADDRESS Defense Advanced Research Projects Agency 1400 Wilson Boulevard Arlington, Virginia 22209		(11) 12. ~~REPORT DATE~~ Novem~~ber 1~~979
		13. NUMBER OF PAGES
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) (12) 187		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Digital Image Processing, Image Understanding, Scene Analysis, Edge Detection, Image Segmentation, CCD Arrays, CCD Processors

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This document contains the technical papers and outlines of semi-annual progress reports presented by the research activities in Image Understanding sponsored by the Information Processing Techniques Office; Defense Advanced Research Projects Agency. The papers were presented at a workshop conducted on 7-8 November 1979 in Los Angeles, California.

407 154 Gm

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

TABLE OF CONTENTS

TABLE OF CONTENTS (Cont.)

FORWARD

The Tenth Image Understanding Workshop under the sponsorship of the Defense Advanced Research Projects Agency (DARPA) convened at the Davidson Conference Center for Continuing Education, University of Southern California, on the 7th and 8th of November 1979. Lt. Col. Larry E. Druffel, Program Manager for the I.U. Research Project in the Information Processing Techniques Office of DARPA welcomed the government officials and research personnel to the two day workshop. Lt. Col Druffel noted that plans are progressing for a demonstration system to evaluate the maturity of IU technology by automating mapping, charting and geodesy functions. While focussing on specific cartographic photointerpretation functions, the system should offer the entire image exploitation community an opportunity to assess the future application of Image Understanding methodologies to their specific problem.

This workshop, which marked the beginning of the fifth year in this research program, proceeded essentially in the pattern established by the previous semi-annual meetings. As usual, the University and Industrial Research Personnel informed the attending representatives from the various Army, Navy, Air Force and Government Agency Organizations about various technical facets of the research effort and provided an overview of progress in Image Understanding Research during the past six month period. An unusual feature of this workshop was a session devoted entirely to the subject of "Symbolic Representation". This session was designed to assist the research community to thoroughly review their opinions on this topic which is key to Image Understanding and to enable those attending to secure a comprehensive overview of the subject. At all sessions, an interchange of views between the researchers and user communitees helped to foster the basic aim of these workshops - improved communication between designers and users.

The papers contained in these proceedings represent the work of the DARPA sponsored research programs at the various institutions involved. Most of the technical papers were presented by the authors at the workshop. However, in the interest of limiting the technical sessions, a few papers are reporduced here which were not able to be formally presented. The Principal Investigators reports included herein are designed to provide a brief outline of the subjects presented by the various P.I.'s. In addition to the technical sessions, participants visited the image processing laboratories of the University of Southern California for live demonstrations of the USC Image Processing Institute capabilities.

The host for the workshop was Dr. Alexander A. Sawchuk, Associate Professor of Electrical Engineering and Director, Image Processing Institute of the University of Southern California. The sponsors and workshop organizers are indebted to Dr. Sawchuk for his untiring efforts in providing arrangements and assuring the success of the workshop. Appreciation is also extended to Ms. Hilda Marti of USC for her work in securing facilities and general assistance and to Ms. Jackqueline Frye of the staff at SAI for typing support for mailings and collection and arrangement of the conference proceedings.

The materials for the cover of this document were provided by Dr. Keith Price, Research Scientist at USC. The layout is designed to show the flow of events as images are processed in the laboratories and are representative of the type of user descriptions which the systems require for processing images of a given area. The semantic network is, by necessity, partial. Many links (relations) have been left out for clarity and relative locations (above, below, etc.) are approximately indicated by the position of the nodes corresponding to the objects. The results of matching this model with the actual image are shown in the San Diego photo on the right side. The photo, Dr. Price tell us, is NASA photo 573001392170. (1703). He notes that this one shows some errors where the ocean is broken apart when the waves become very clear and where some linear segments are clearly misidentified as being part of the major highway. The artwork and layout is the work of Mr. Thomas G. Dickerson of the Art Department of Science Applications, Inc.

Lee S. Baumann
Science Applications, Inc.
Workshop Organizer

AUTHOR INDEX

SESSION I

TECHNICAL PAPERS

DEVELOPMENT OF CUSTOM-DESIGNED INTEGRATED CIRCUITS FOR IMAGE UNDERSTANDING

G.R. Nudd, S.D. Fouse, and T.A. Nussmeier

Hughes Research Laboratories, Malibu, California 90265

and

P.A. Nygaard

Carlsbad Research Center, Carlsbad, California 92008

ABSTRACT

This paper describes our on-going program to develop special-purpose charge-coupled device and metal oxide semiconductor integrated circuits for real-time image processing. This work has emphasized the development of circuits that will perform the front-end, or "low-level," processing functions at data rates in excess of 10^6 pixels/sec. We describe the design and fabrication of a third test chip which will perform two-dimensional processing operations over kernel sizes ranging from 3x3 to 26x26 pixels. Included on this chip are data programmable operations for processing over a 5 x 5 kernel at real-time television rates. In addition, we describe the test facilities we have designed and built to demonstrate the performance of these circuits and the initial test results.

I. INTRODUCTION

A primary aim of the program has been to demonstrate the feasibility of performing image-understanding algorithms in real time. For our purposes, we define "real time" to be equivalent to high-quality television, 7.5-MHz data rate. Even for relatively simple operations on kernels of 3x3 or 5x5 pixels, this represents a speed increase over conventional general-purpose computers of at least two or three orders of magnitude. To achieve this increased throughput, we have designed and implemented novel charge-coupled device (CCD) and metal oxide semiconductor (MOS) processing architectures that can be integrated into infrared and video cameras. Based on the work funded on this program, we are currently investigating several military applications that require both the sensor and processor to be integrated onto a single chip (the so-called "smart-sensor" philosophy).

We have designed, built, and tested three integrated-circuit test chips containing the 14 algorithms listed in Table 1. Each chip has been used to demonstrate a different approach to image analysis. The first chip shown in Figure 1 was aimed at demonstrating a novel two-dimensional CCD filtering approach which allows concatenation of several (in this case 5) image-understanding operations and has been designed to be integrated into the sensor directly at the focal plane. Each of the functions on this chip operates over a 3x3 array of picture elements and provides a single processed picture element for each new input pixel. The circuit accepts three lines of video data equivalent to the 3x3 array and as such requires two external analog delay lines when operated from a vidicon or commercial camera, as shown in Figure 2. In our initial work to operate these

7819-10 R1

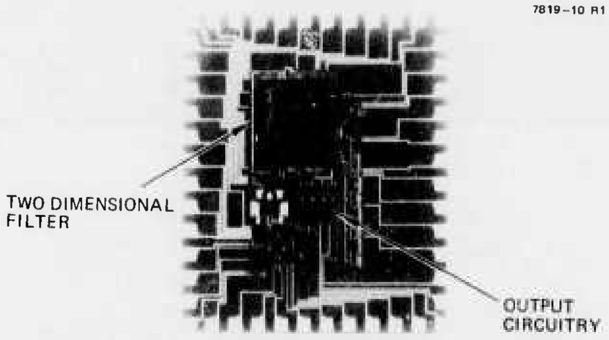


Fig. 1. Photomicrograph of test chip I.

9136-2

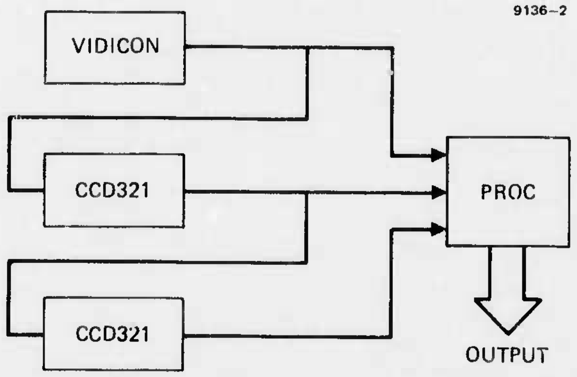


Fig. 2. Formation of the 3x3 pixel array using external analog delay lines.

Table 1   Integrated Circuit Primitives Developed to Date

9136-1

TEST CHIP NUMBERS	ALGORITHMS IMPLEMENTED	KERNEL SIZE	OPERATIONS PER PIXEL	PERFORMANCE			STATUS
				NUMBER OF BITS	SPEED (PIXEL RATE)	EFFECTIVE OPERATION RATE	
I	EDGE DETECTION	3 x 3	16	4	5 kHz	80 KOPS	DEVELOPED AND TESTED
	HIGH-PASS SPATIAL FILTER	3 x 3	18	4	5 kHz	90 KOPS	
	LAPLACIAN	3 x 3	13	4	5 kHz	65 KOPS	
	12 dB/APERTURE CORRECTOR	3 x 3	18	4	5 kHz	90 KOPS	
II	SOBEL	3 x 3	16	4	2 MHz	32 MOPS	DEVELOPED AND TESTED
	MEAN	3 x 3	9	4	2 MHz	18 MOPS	
	UNSHARP MASKING	3 x 3	13	4	2 MHz	26 MOPS	
	BINARIZATION	3 x 3	10	4	2 MHz	20 MOPS	
	ADAPTIVE STRETCH	3 x 3	12	4	2 MHz	24 MOPS	
III	LAPLACIAN	3 x 3	13	6^+	$7 MHz^+$	$91 MOPS^+$	IN PROCESS
	MASK PROGRAMMABLE	7 x 7	98	6	7 MHz	536 MOPS	
	PROGRAMMABLE	5 x 5	50	6	7 MHz	350 MOPS	
	PLUS SHAPED MEDIAN			6	7 MHz		
	BIPOLAR CONVOLUTION	26 x 26	1352	6	7 MHz	$\sim 10^4$ MOPS	

$^+$PREDICTED PERFORMANCE BASED ON DESIGN

circuits in real time, we have used Fairchild CCD 321 analog line delays. These have worked quite well but limit both the dynamic range and the signal-to-noise ratio of the processor. We are currently investigating techniques for incorporating the circuits directly into a CCD imager as shown in Figure 3.

The second chip, shown in Figure 4, contains five individual circuits again using a 3x3 pixel kernel and is aimed at demonstrating adaptive processing using the local mean as the control. This circuit has been operated directly from both a vidicon and CCD camera with an overall processing accuracy equivalent to 4 bits. The operations performed in addition to the 3x3 average are adaptive stretch, binarization based on the local mean, unsharp masking, and Sobel edge detection. Under a parallel contract with Night Vision Laboratories, Fort Belvoir, Virginia, we have integrated these circuits into a demonstration processor, shown in Figure 5. At the request of the customer, we incorporated a CCD field delay to remove the interface and provide a processing capability on adjacent lines of video. This processor has been operated at a 4-MHz clock rate, and the results are reported in Ref 1.

9136-3

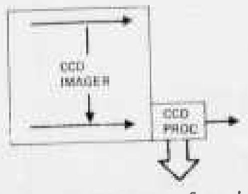


Fig. 3. Technique for integrated CCD images and processor

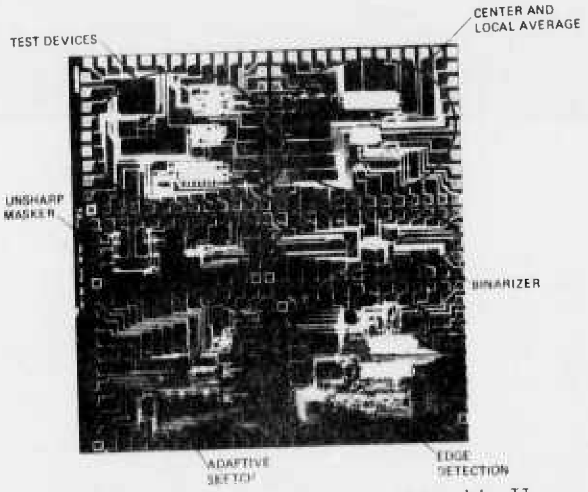


Fig. 4. Photomicrograph of test chip II.

Our recent work has been concerned with the design, processing, and initial evaluation of a third test chip, which is aimed at demonstrating processing techniques using larger kernel sizes (as high as 26x26 pixels) and demonstrating a programmable capability. After many problems and delays in obtaining a satisfactory mask set, we have now completed the processing of this chip and are currently collecting preliminary performance data on each of the five circuits, as described below.

II. PROGRESS ON TEST CHIP III

The principal effort this period has been with the design, simulation, and processing of this

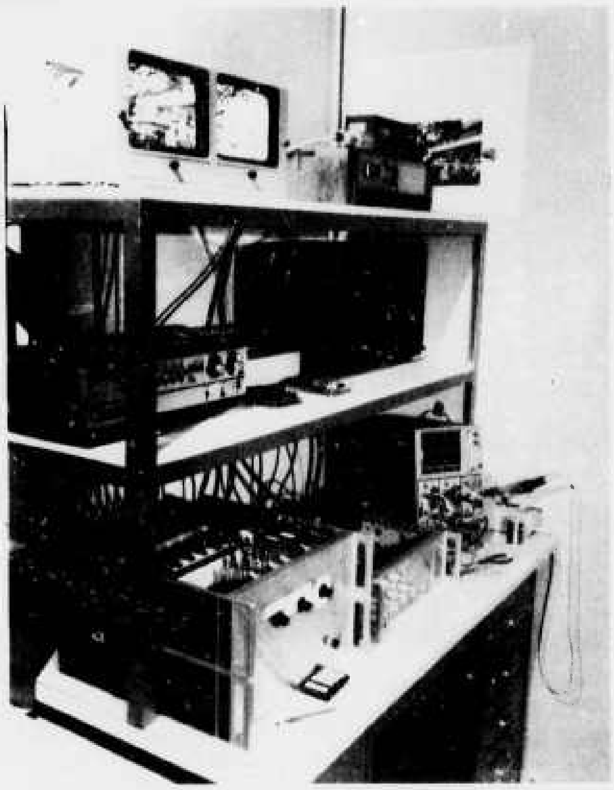


Fig. 5. Real-time test facility for test chip II.

chip. Five functions, a 7x7 mask programmable array, a 3x3 Laplacian, a median operator, a 5x5 voltage programmable convolution, and a large 26x26 element convolution for the primal sketch,[2] are included. The design goal is for a 15-MHz clock rate and an overall processing accuracy equivalent to 6 bits. The resolution of the circuit lithography is 5 to 7 μm, equivalent to commercial optical techniques, and the technology is n-type surface channel. The bandwidth requirements for this chip are towards the high end of the speed capability range for surface channel devices and hence represent a considerable challenge. Also, the kernel size has been considerably extended from the 9 pixels used in our previous work. The largest processor, the convolution for the primal sketch, contains 338 pixels. Further, the dynamic range required by the operators contained on the chip is much increased, representing approximately 8 bits, and we are including special techniques to achieve this. Probably the most significant challenge we are addressing on this chip is the development of programmable processing kernels. The concept here is to develop a general-purpose convolutional processor that can accept data at real-time video rates and can adapt its kernel size and weights either in a preprogrammed way or in response to the processed output at a speed higher than the frame rate (>30 Hz). If such a device can be developed with accuracy equivalent to 6 bits, it will find very widespread general utility in image analysis and understanding. At present, the kernel size for this circuit is 5x5, but there is no fundamental limit preventing this from being significantly increased. To meet the significantly increased demands both in terms of speed and dynamic range, we have included an on-chip sample and hold to both reduce the output noise and lower the clock feedthrough. This should significantly increase the performance of the functions, particularly at high speed. A schematic of the circuit is shown in Figure 6. This device has been simulated to operate at an 11-MHz data rate and provide a 60-dB common mode rejection driving a 30-pF output load. This circuit is included in each of the CCD functions.

A photomicrograph of the full chip showing each circuit and the test devices is shown in Figure 7. The chip itself is approximately 225 mils^2, which is slightly larger than our previous one (191 mils^2). This has resulted in fewer dice per wafer, thus requiring higher yield to provide acceptable quantities for testing. We are currently processing 11 wafers, each with 36 dice/wafer. This should hopefully result in enough acceptable circuits for initial testing. Later we will process an additional lot when the chip's initial operating parameters have been determined.

This chip has now been designed and simulated using the circuit analysis and simulation program SPICE. The individual cells have been drawn and the composite digitized by the mask maker. Because of high demand in the IC market, the turnaround at the mask maker has been somewhat longer than originally anticipated. In addition to this delay, the initial mask set received was incomplete; also, some of the masks were reversed field. The net effect of these two delays is an anticipated schedule slippage of several months. We have only recently completed the processing required prior to short testing the devices, dicing the wafer, and packaging. However, this process is now complete, and we have started the initial performance evaluation. Because of the larger kernel size and the significantly different characteristics of these circuits, we essentially have had to rebuild the test facility. Considerable time and effort was expended on this during this period, as discussed below.

III. PERFORMANCE, EVALUATION, AND TESTING

To be able to effectively test the performance of the microelectronic circuits developed, it has been necessary to develop an appropriate test facility. The essential components of this system are shown in Figure 8. The functions developed on the test chip include both two-phase and single-phase circuits, each requiring a clock swing of at least 20 V. To achieve this, we have developed a driven system that operates from an 8-channel (T^2L voltage level) word generator and uses these outputs to develop MOS level waveforms. This generator is also used to develop the necessary diffusion and reset pulses. The word generator can be clocked at a 20-MHz bit rate to process a 3-MHz bandwidth signal. The imagery will be obtained either from a stored data base or a vidicon camera.

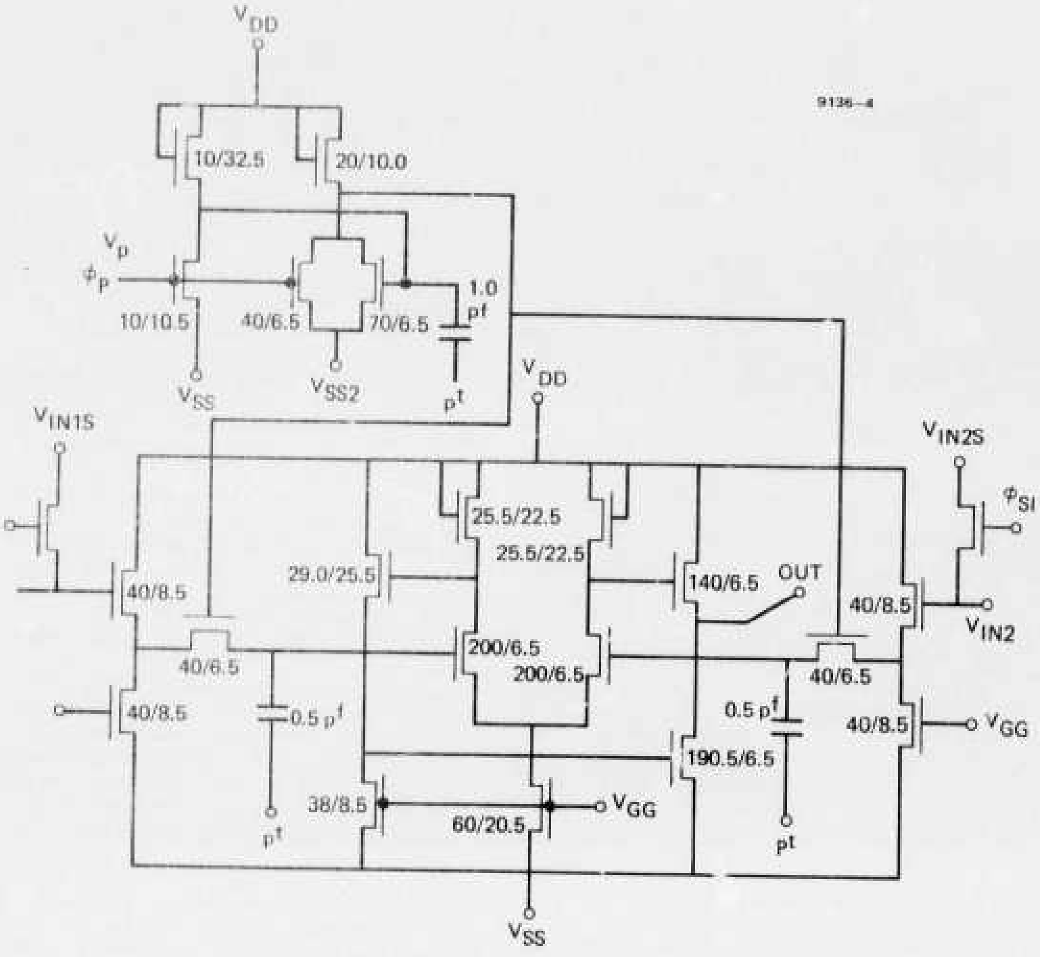


Fig. 6. Schematic of on-chip sample and hold

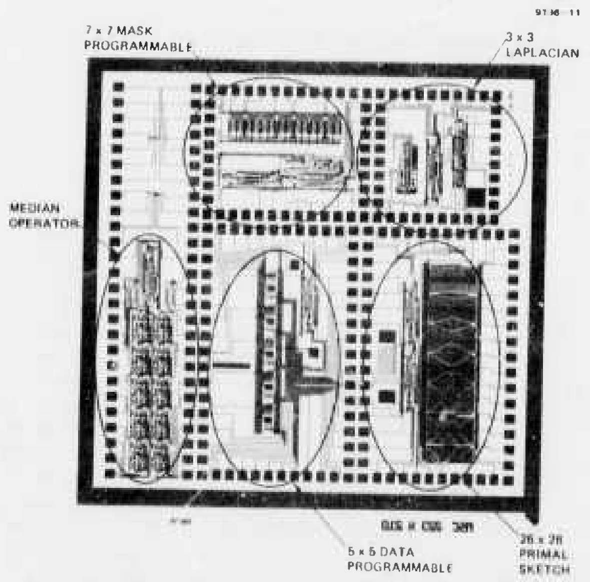


Fig. 7. Photomicrograph of image understanding chip III.

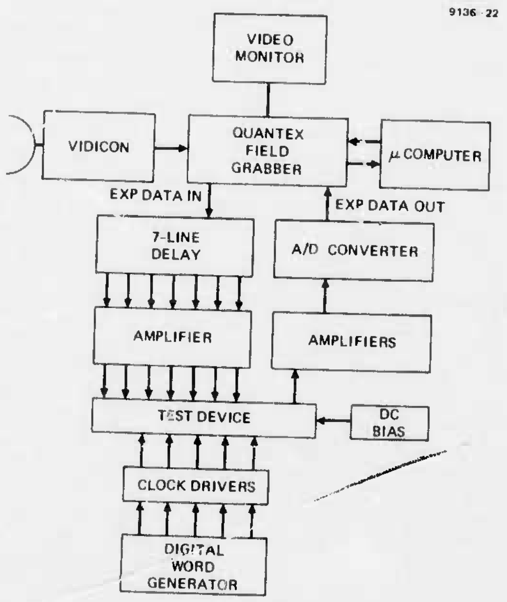


Fig. 8. Schematic of test facility.

In either case, the data requires formatting into several parallel video lines to form the appropriate kernel size. The system to do this has now been completed. It consists of a 10-MHz, 8-bit analog-to-digital converter system that provides input to a 24-kbit RAM register. The RAM register provides a delay equivalent to six horizontal lines. A digital-to-analog converter is included after each 4 kbits to provide the analog output from the adjacent lines. The necessary hardware to provide this facility has now been completed and the system tested with a commercial vidicon. In addition, a signal conditioner box, which both translates the dc level of the resulting video data and can provide the necessary variable gain, has been designed and built. We also have provided the capability to vary both the spatial and temporal resolution of the processor and have investigated several commercial "frame grabbers" and digital memories to provide this facility. The system we built is based on the Quantex Field Grabber with resolution of 256x256 pixels each with 6 bits of gray scale. We have interfaced this system to an IMSAI 8080 microcomputer for evaluation and have written several software packages to manipulate the data to provide both simulation of image-understanding operations and manipulation for display purposes.

We are currently using this facility to evaluate the first lot of test wafers. We have decided to investigate the 5x5 data programmable convolution and the median operator initially. The schematic of the 5x5 programmable operator is shown in Figure 9. Essentially, it accepts five parallel lines of video data and performs a 25-point bipolar convolution on a sliding 5x5 pixel array. The mathematical formulation of the processor is given by

$$\hat{I}_{k,l} = \sum_{i=k-2}^{k+3} \sum_{j=l-2}^{l+3} I_{i,j} \cdot W_{ij} \quad ,$$

where $\hat{I}$ is the intensity of the processed image, I is the original image, and the W_{ij} are the programmable weights. The processor consists of a two-dimensional floating gate array with 25 voltage-controlled taps. This array overlays five separate CCD delay lines through which charge equivalent to each picture intensity is clocked. A single source follower at the end of all of the delay lines is used to detect the linearity and transfer efficiency of the basic CCD structure. An example of the operation in this mode is given in Figure 10. The input signal is shown on the upper trace, and the output resulting from two cycles of this waveform is shown on the lower trace. The operator here is at ∿6 kHz, equivalent to the pixel rate of the stored data base microprocessor system. The photograph illustrates the need for an output stage including a sample and hold. The reset pulses occurring each pixel interval can be seen to have about the same magnitude as the diffusion output. These are due entirely to feedthrough and are unwanted outputs. To avoid this, we have tested the on-chip sample and hold as shown in Figure 11. In this case, the input signal consists of a single frequency that is sampled each 60 μsec. At these intervals, the output level of the waveform is sampled and frozen until the next sample pulse. This circuit is included on all the active CCD outputs and is used to eliminate the unwanted clock to reset feed-throughs. The initial results obtained from the floating gate are shown in Figure 12(a and b). In each case, the input signal is shown on the upper

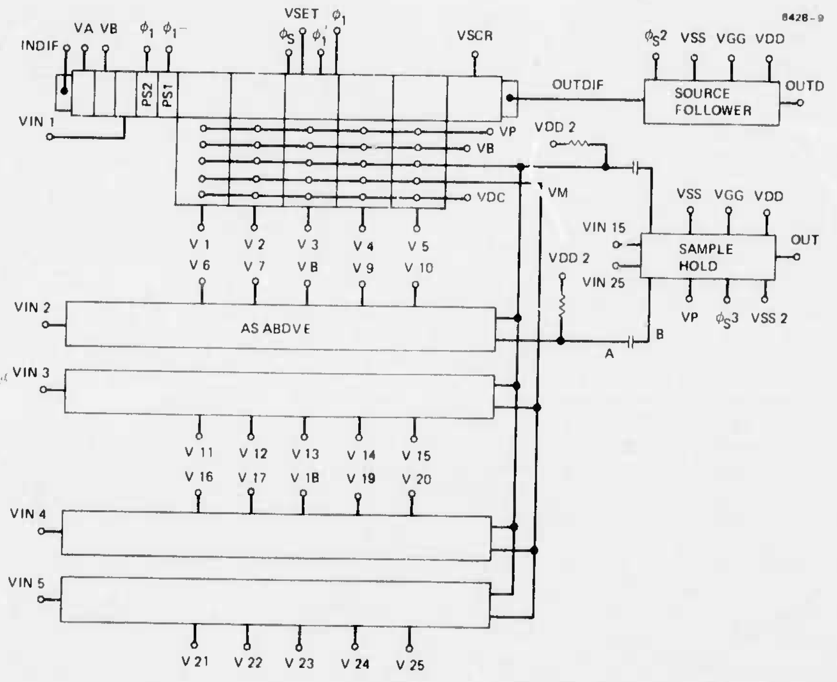


Figure 9. Schematic of 5x5 programmable filter.

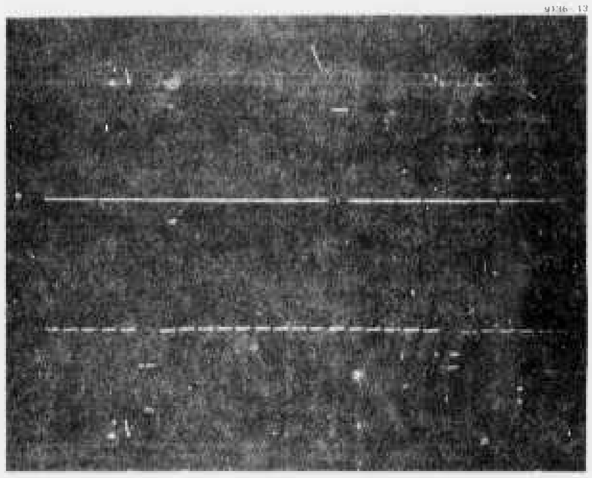


Fig. 10. Output from source-follower on the 5x5 programmable processor.

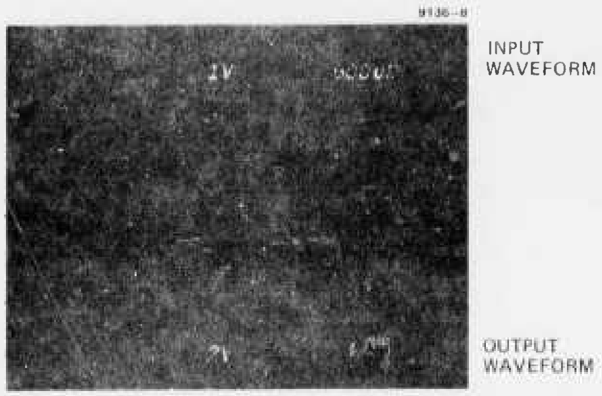


Fig. 11. Operation of the on-chip sample and hold.

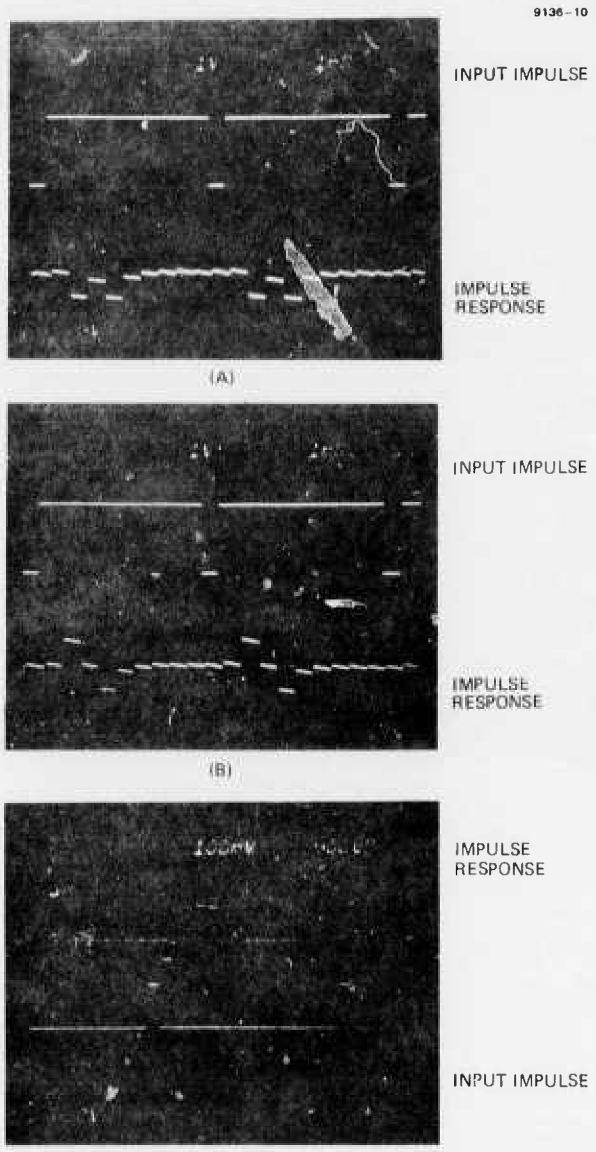


Fig 12. Impulse response of programmable array for setting (0, -1, 0, -1, 0), (0, 1, 0, -1, 0), and (1, 1, 1, 1, 1).

trace and the resulting impulse response shown below. In Figure 12(a) the voltage settings on the floating arrays are equivalent to 0, -1, 0, -1, 0, and in Figure 12(b) to 0, 1, 0, -1, 0. In each case, the input response equivalent to this setting is obtained, verifying both the programmability and the capability of bipolar weighting. Figure 12(c) shows the impulse response equivalent to 1, 1, 1, 1, 1. Although these are preliminary results, it is clear that all the necessary functions are operating on the chip. The dynamic range of the device obtained so far is illustrated in Figure 13. It is equivalent to about 14 gray levels. The granularity on the trace shown is due in part to the microprocessor system used to obtain this transfer characteristic. Our work will continue on this circuit to improve both the accuracy and dynamic range and to increase the speed to 7.5 MHz. We have, however, started to process some test-patterns, as shown in Figure 14. The first consists of a two-dimensional grid with lines each a single pixel in width. This has been processed with the processor set for a 1, 1, 1, 1, 1 impulse response in both x and y. The resulting grid with each line 5 pixels wide is shown in Figure 14(b). This corresponds directly to the simulation shown in Figure 14(c). A similar result for the picture of the house is shown in Figure 15. The impulse responses for the programmable array setup to be both a plus shape and a cross shape are shown in Figure 16.

During the next phase of the program, we will continue the testing and evaluation of the circuits and provide a detailed performance review.

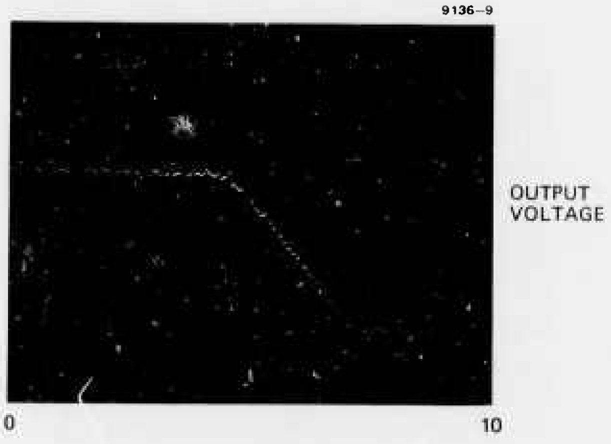


Fig. 13. Linearity and dynamic range of 5x5 programmable processor.

IV. STUDY OF VLSI FOR IMAGE UNDERSTANDING

In addition to the work on the CCD/MOS circuitry and building a new test facility, we have begun an analysis program to determine the impact and advantages of very large scale integration (VLSI) on image understanding. The on-going work in software and algorithm development clearly is leading to an ever-increasing demand for computational throughput. Further, it is evident that, even with the most sophisticated general-purpose machines, the processing times are incompatible with any real-time application. An apparent solution to these issues is the development of new processing architectures based on the latest technologies. Two significant developments have been taking place in microelectronics in the past several years. The first is the development of a variety of new technologies such as DMOS, CMOS/SOS, I^2L, ECL, and GaAs. Each offers a different combination of parameters in terms of speed, power consumption, and the possible level of integration. It is of particular importance to the I.U. program to be able to evaluate the advantages and constraints of all the available technologies with respect to our programs. In addition, with increased resolution of integrated-circuit features and decreased power consumption, the level of integration within each chip has greatly increased. For example, in highly regular arrays, such as memories, as many as 10^5 to 10^6 functions can be integrated in each chip. This development will also have profound effects.

To some extent, the design and architecture of high-density circuitry other than memories and commercial microprocessors have lagged behind these developments, and a significant advance can be expected from the optimum design of image-processing architectures. To achieve the maximum advantages of VLSI for the I.U. problem, two precepts must be adhered to. First, the designs and architectures must, where possible, provide concurrent operation. The obvious bottlenecks that result in the von Neuman concepts of a single arithmetic unit, etc. can be circumvented by highly

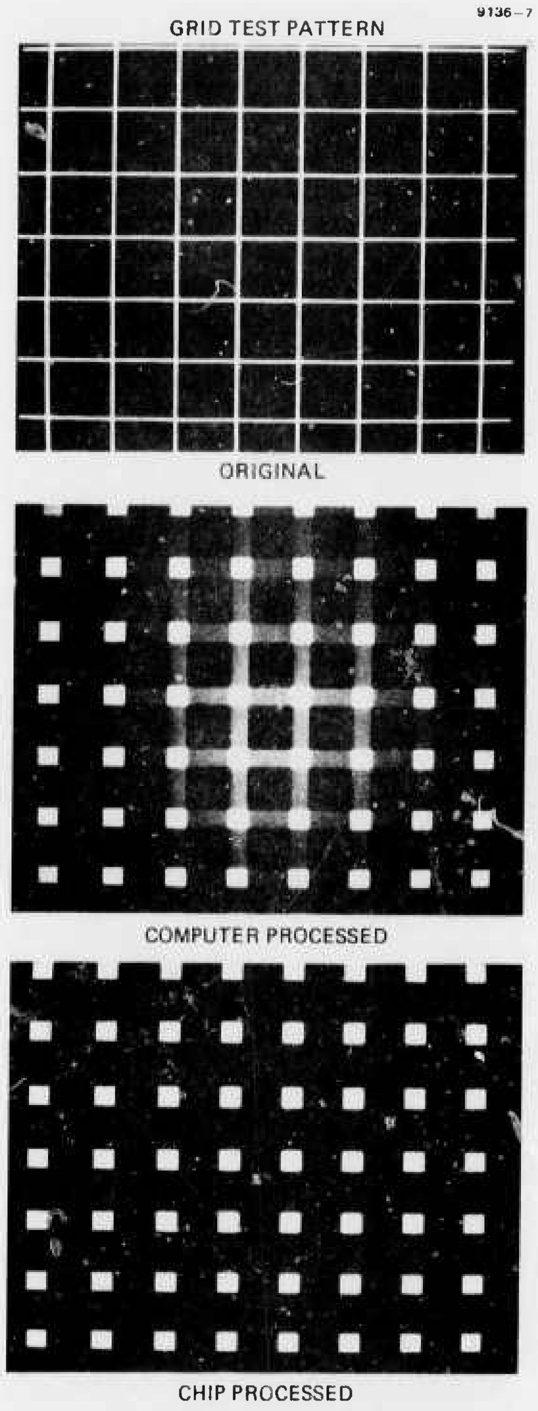


Fig. 14. Performance of 5x5 programmable processor on test pattern. (all weights equivalent to unity)

localized operation with multiple primitives. This removes many of the problems from the processor and places them in the control and data distribution system. There is, for example, a need for localized distributed storage and memory associated with

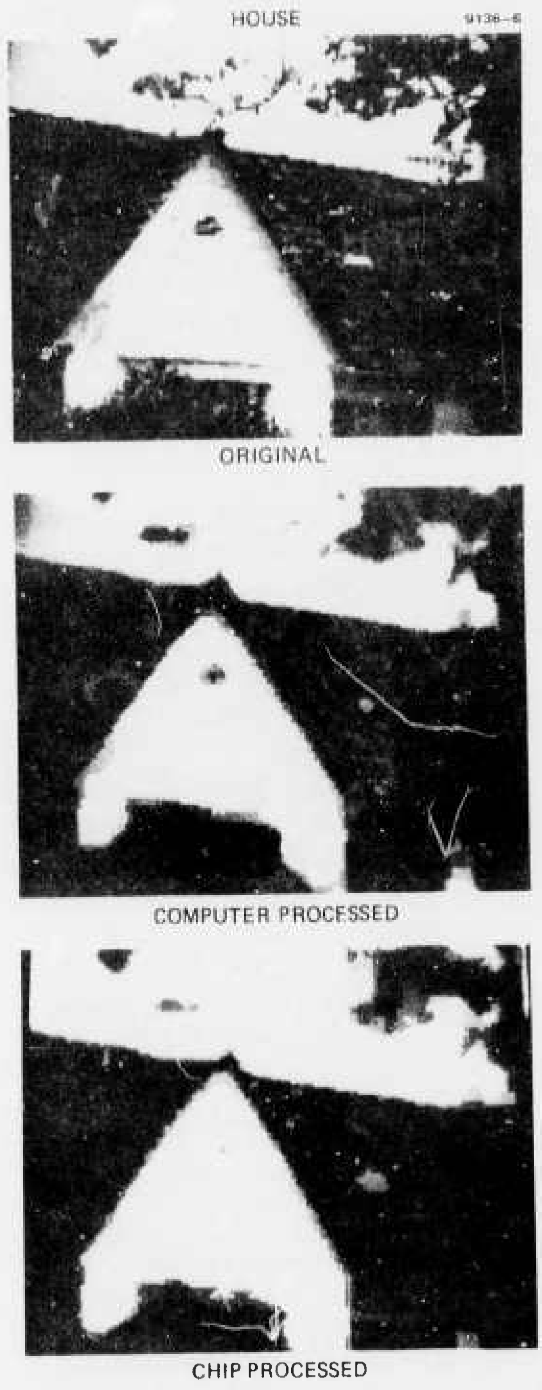


Fig. 15. Example of processor used on stored imagery (settings as for Fig. 14).

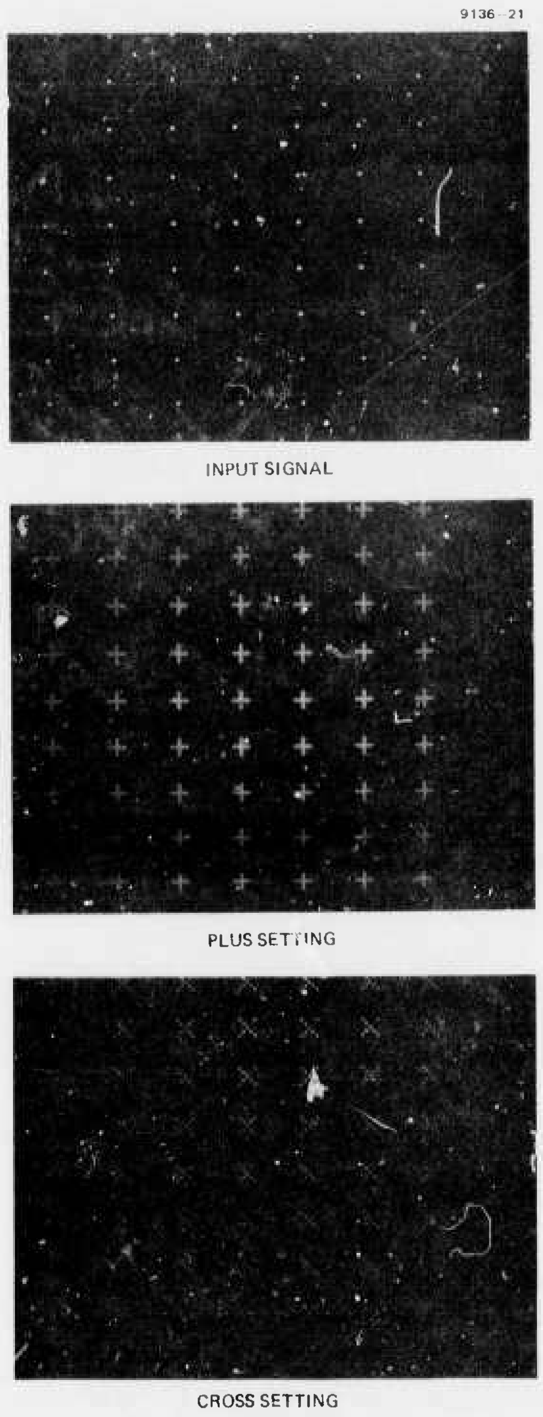


Fig. 16. Impulse response of programmable filter.

each primitive for data and instruction queuing. In addition, both for ease of design and for increased packing density, the circuitry must be highly regular on the silicon surface.

From our previous work on this program, it is clear that, using current state-of-the-art technology, low to intermediate level primitives can be built that will provide real-time operation without requiring extensive area on the silicon. From this work, we anticipate that five or six primitives might be included in a single 200-mil^2 chip. In support of this concept, we are currently investigating concepts for data distribution and intelligent local storage as well as techniques such as residue operation and number theoretical transformation for regularizing the processes themselves. The eventual aim of this work, which will continue in the next period, is to determine an optimum way of mapping the algorithms and processors onto the two-dimensional silicon.

V. SUMMARY AND FUTURE PLANS

During the current period, our work has been concentrated in three areas: the fabrication of Test Chip III, the development of an effective test facility and preliminary testing of the circuits, and an initial study of the effect of VLSI on image understanding. Progress in each of these areas has been satisfactory, although unexpected delays have been encountered primarily with the outside mask maker. The problems with this vendor have created an unavoidable delay of at least two months and have been largely responsible for our delay in the testing schedule. However, all 11 masks have now been received, and the wafers have been processed and short tested. We have completed nearly all the work on the test facility and are now getting initial test results. This work will continue next period, and we intend to issue an interim report detailing the test results as available. In addition to this, we have started our VLSI study and are starting to formulate an approach for this major task next year.

REFERENCES

1. A Charge Coupled Device Image Processor for Smart Sensor Applications, G.R. Nudd, G.D. Thurmond, S.D. Fouse, S.P.I.E. Conference, San Diego, August 30-31, 1978, S.P.I.E. Vol. 155, pp. 15-22.

2. D. Marr and T. Poggio, A Computational Theory of Human Stereo Vision, M.I.T. A.I. memo 451 (1977).

INVESTIGATION OF VLSI TECHNOLOGIES FOR IMAGE PROCESSING

W.L. Eversole, D.J. Mayer, F.B. Frazee, and T.F. Cheek, Jr.

Texas Instruments Incorporated
13500 North Central Expressway, P.O. Box 225936
Dallas, Texas 75265

ABSTRACT

This paper summarizes recent work performed under a subcontract from Carnegie-Mellon University for the DARPA Image Understanding Program. Discussion of the implementation of a real time median operator and a programmable sum of products operator are presented.

I. INTRODUCTION

The concept of very large scale integration (VLSI) implementation of a real time digital image processor based on multiple arithmetic logic units (ALUs) and on-chip buffer memories was presented at an earlier workshop.[1] The implementation of the appropriate buffer memories was discussed at the last workshop.[2] While recent advances in component technology now make possible the realization of real-time image processors capable of performing highly complex functions, an understanding of the potential for implementing complex algorithms with miniaturized hardware is a necessary tie between algorithm research and hardware development efforts. The need to properly define the complex functions before actual integrated circuit design begins is imperative due to the complexity of image processing algorithms and the development cost and schedules of an integrated circuit design. A poorly defined function or a hastily made technology decision can destroy an otherwise successful program. The design and implementation of a breadboard version of a proposed integrated circuit is considered good engineering practice. The breadboard version allows evaluation of the algorithm as well as the discovery and evaluation of the problems, risks and options involved before incurring enormous expenses.

This paper discusses the breadboard versions of two image processing algorithms; a 5 x 5 median of medians operator, and a programmable sum of products operator.

II. MEDIAN OF MEDIANS OPERATOR

The 5 x 5 median of medians[1] operator discussed at a previous workshop was implemented using off-the-shelf components. The breadboard will allow evaluation of the median of medians operator and provide important design inputs for a completely integrated version. A block diagram of the breadboard is shown in Figure 1. The breadboard was designed for real-time operation using a commercial TV camera as the sensor. Commercially available 8 bit analog-to-digital and digital-to-analog converters were used. The memory needed to buffer 4 lines of video was implemented using 1024 x 4 bit static random access memories (RAMs). The median operator circuitry was implemented using low power Schottky transistor-transistor logic. The ability to operate on images of different resolutions was accomplished by controlling the timing of the median board to accept every output sample of the buffer memory for full resolution or every other sample for 1/2 resolution, every fourth sample for 1/4 resolution or every eighth sample for 1/8 resolution. The power required for the buffer memory function and the median of medians operator is 10 watts and 14 watts, respectively. The power and size of these simple functions emphasize the need for integrated circuit technology in the implementation of image processing functions.

The operation of the median of medians breadboard is shown in Figure 2. The original image is corrupted by impulse-like noise. The noise is seen as small dark areas on the face of the girl. After passing through the median of medians breadboard the impulse noise is removed. The median filtered image appears somewhat blurry at sharp edges of the image. This is caused by the signal suppression property of the two dimensional 5 x 5 median filtering. Operation of the median of medians breadboard at different resolutions is shown in Figure 3. As the resolution is decreased the median filtered image becomes less recognizable.

The median of medians breadboard has been delivered to Carnegie-Mellon University along with a simple interface board for interfacing to the VAC 11/780 computer. This will allow the 5 x 5 median of medians filter to operate on images up to 1024 x 1024 pixels.

III. PROGRAMMABLE SUM OF PRODUCTS OPERATOR

Many image processing algorithms require operations of the form

$$y = \sum_{i=0}^{M-1} W_i X_i \qquad (1)$$

where the W_i's represent a set of fixed weighting coefficients and the X_i's represent a set or sequence of input values. Equation (1) can be used to calculate the coefficients of various transforms such as Fourier, Cosine, Hadamard, Haar, etc. Where two dimensional transforms are needed, successive one dimensional transforms can be used if the transforms are separable. Another very important application of Equation (1) is the discrete convolution of a two dimensional input image with a convolution array. These mathematical operations based on the neighboring pixel values are termed neighborhood operators and are used in many image processing algorithms. Examples of neighborhood operators include noise smoothing, edge crispening, linear edge enhancement, etc.

Equation (1) can be implemented using digital multipliers and adders; howerver, the size and power required to perform the multiplication at video data rates with the accuracy needed for most image processing applications is prohibited.

A technique for realizing Equation (1) that does not require digital multiplication is the ROM-accumulator (RAC)[3] technique. The RAC technique implements the sum of products of an input word set with a set of weighting coefficients using a table look-up procedure as discussed at the last workshop.[2]

To properly define the LSI/VLSI implementation of the programmable sum of products operator, a breadboard version of a 3 x 3 programmable operator using off-the-shelf components is being designed and implemented. A block diagram of the 3 x 3 programmable operator breadboard is shown in Figure 4. The breadboard consists of nine input latches, nine parallel in-serial out shift registers, a fast 512 x 12 bit memory for temporary storage of the partial products, an EPROM for permanent storage of the partial products, shift and accumulate circuitry, tri-state output latches, and control circuitry.

The input latches are used to buffer the 8-bit input data paths (A, B and C) into the RAC circuitry. They also act as an 8-bit wide shift register for sliding window operations. When operating on 3 x 3 pixel blocks, data paths A, B and C are applied in parllel to latches L2, L5, and L8, respectively by appropriate control of the 2 multiplexers. In this case, the input latches act as 3 parallel, 3-pixel shift registers. When operating on 9 x 1 pixel blocks, data path C is applied to latch L8 and data paths A and B are not used. In this case the multiplexers allow the input latches to form a single 9-pixel shift register.

When data is valid in the input latches, their contents are clocked in parallel into the parallel in-serial out shift registers. These registers convert each bit-parallel input word into bit-serial form which provides 8 sequential 9-bit addresses to the partial product memory. This memory is composed of 12, 30 ns static 1K x 1 MOS RAMs. The desired partial products stored in this memory are initially downloaded from EPROMs or alternatively from a computer.

The 8 sequential partial products obtained from the memory are applied to the shift and accumulate circuitry which performs binary weighting and summation using carry-save addition techniques.[4] Unsigned mangitude or two's complement data may be used. The accumulator output is latched in tri-state buffers.

Using commercially available components the maximum internal clock rate for the shift registers, memory, and accumulator is 20 MHz. Thus, for 8-bit wide input data, 400 ns (8 x 50 ns) are required to complete each sum of products calculation. When performing 3 x 3 sliding window operations, the maximum input date rate is therefore 2.5 MHz. In order to perform 3 x 3 sliding window operations at standard TV data rates, three boards may be operated in parallel with 6-bit input data to achieve an input data rate of 10 MHz. When performing 9 x 1 non-sliding operations with a single board, the maximum 8-bit input data rate is theoretically 22.5 MHz. The breadboard, however, is designed for 10 MHz maximum input data rate.

The design of the breadboard is complete and fabrication has begun.

CONCLUSIONS

This paper has discussed the implementation using off-the-shelf components of two image processing functions; a median of medians operator and a programmable sum of products operator. These breadboard activities are invaluable aids to integrated circuit architecture design. Experimental results from the median of medians operator were shown and a detailed discussion of the programmable sum of products breadboard was presented. Work is continuing on the fabrication of the programmable sum of products breadboard.

REFERENCES

1. W.L. Eversole, D.J. Mayer, F.B. Frazee and T.F. Cheek, Jr., "Investigation of VLSI Technologies for Image Processing," Proc. Image Understanding Workshop, Pittsburg, Penn., November 1978, pp. 191-195.

2. W.L. Eversole, D.J. Mayer, F.B. Frazee and T.F. Cheek, Jr., "Investigation of VLSI Technologies for Image Processing," Proc. Image Understanding Workshop, Palto Alto, California, April 1979, pp. 159-163.

3. T.A.C. Classen, W.F.G Mecklenbrauker, and J.B.H. Peek, "Some Considerations on the Implementation of Digital Systems for Signal Processing," Phillips Research Reports, Vol. 30, pp. 73-84, 1975.

4. H.J. DeMan, C.J. Vandenbulcke, and M.M. Van Cappelen, "High Speed NMOS Circuits for ROM-Accumulator and Multiplier Type Digital Filters," IEEE Journal of Solid State Circuits, Vol. SC-13, pp 565-572, October 1978.

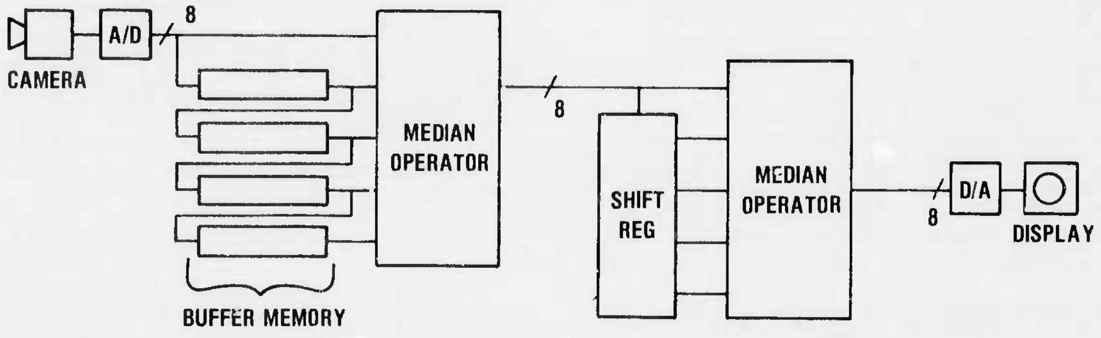


FIGURE 1. BLOCK DIAGRAM OF 5x5 MEDIAN OPERATOR BREADBOARD

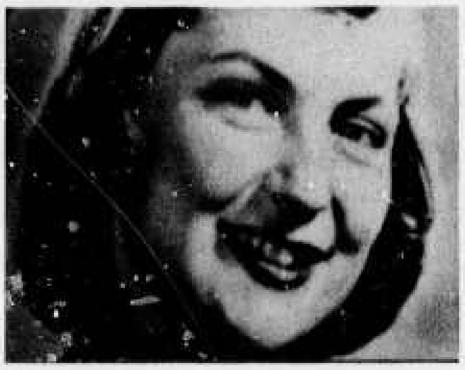
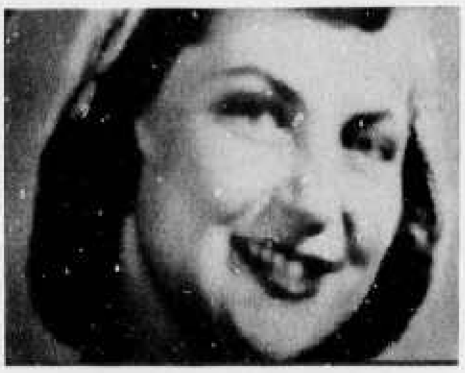


IMAGE WITH IMPULSE NOISE



MEDIAN FILTERED IMAGE

FIGURE 2. OPERATION OF 5x5 MEDIAN OF MEDIANS BREADBOARD

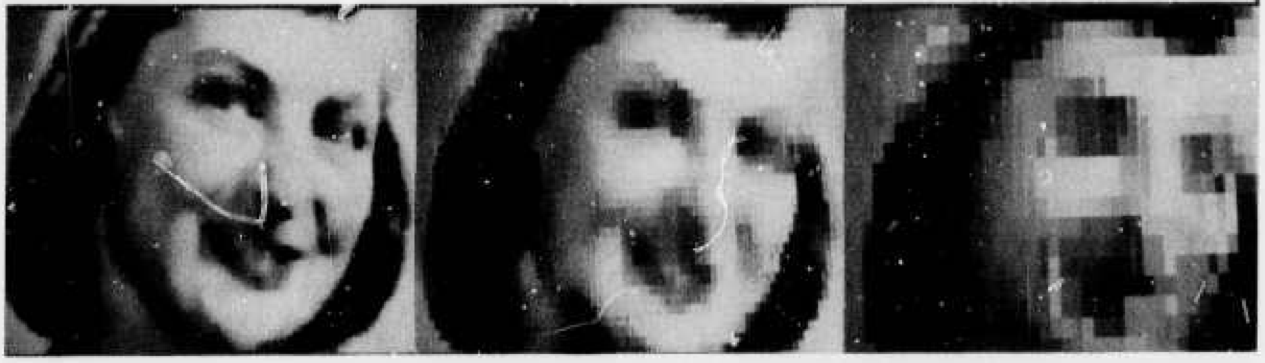
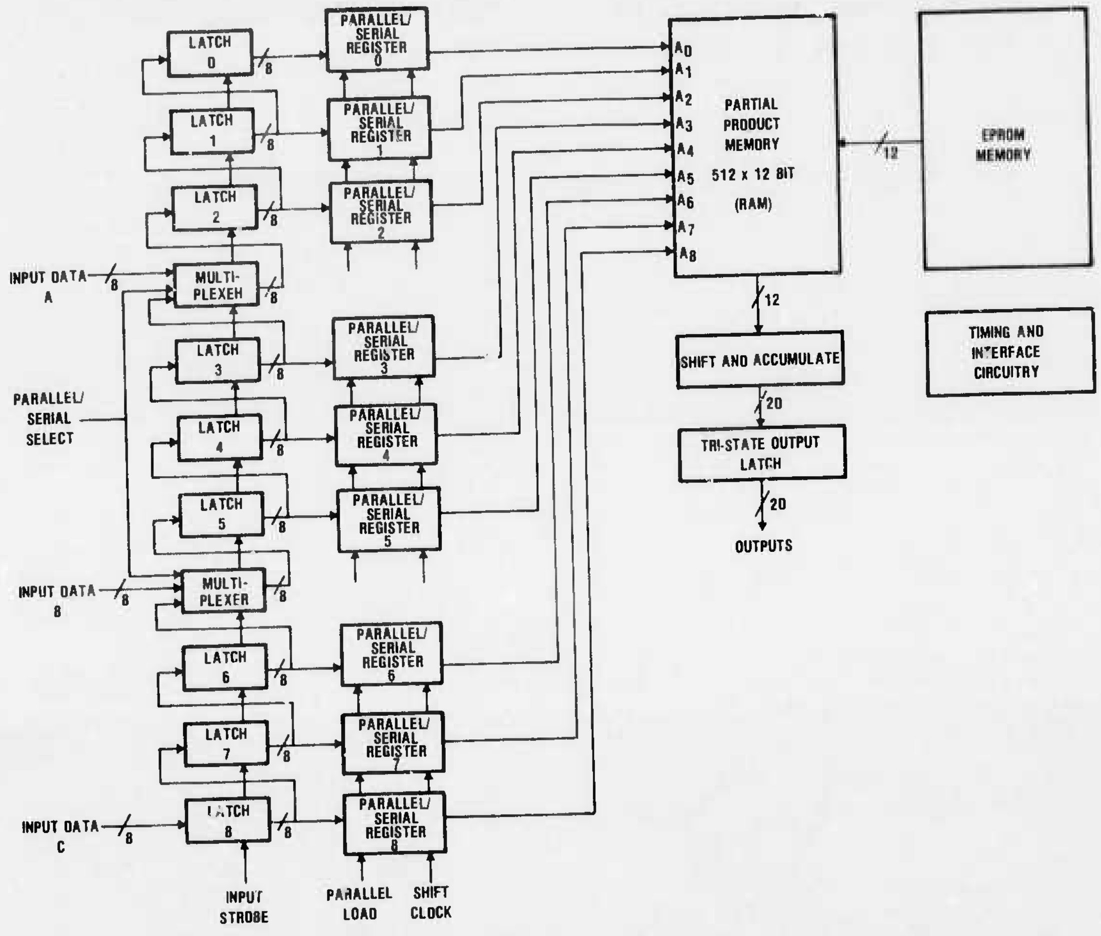


FIGURE 3. OPERATION OF 5x5 MEDIAN OF MEDIANS BREADBOARD
WITH DIFFERENT RESOLUTIONS



LATCH 0
LATCH 1
LATCH 2
LATCH 3
LATCH 4
LATCH 5
LATCH 6
LATCH 7
LATCH 8
MULTI-PLEXER
MULTI-PLEXER
PARALLEL/ SERIAL REGISTER 0
PARALLEL/ SERIAL REGISTER 1
PARALLEL/ SERIAL REGISTER 2
PARALLEL/ SERIAL REGISTER 3
PARALLEL/ SERIAL REGISTER 4
PARALLEL/ SERIAL REGISTER 5
PARALLEL/ SERIAL REGISTER 6
PARALLEL/ SERIAL REGISTER 7
PARALLEL/ SERIAL REGISTER 8
INPUT DATA A
INPUT DATA B
INPUT DATA C
PARALLEL/ SERIAL SELECT
INPUT STROBE
PARALLEL LOAD
SHIFT CLOCK
8
A0
A1
A2
A3
A4
A5
A6
A7
A8
PARTIAL PRODUCT MEMORY 512 x 12 BIT (RAM)
12
EPROM MEMORY
SHIFT AND ACCUMULATE
20
TRI-STATE OUTPUT LATCH
OUTPUTS
TIMING AND INTERFACE CIRCUITRY

FIGURE 4. BLOCK DIAGRAM OF PROGRAMMABLE SUM OF PRODUCTS BREADBOARD

HIGHER LEVEL ALGORITHMS: EVALUATION AND IMPLEMENTATION

Thomas J. Willett
Arden R. Helland
Glenn E. Tisdale

Westinghouse Systems Development Division, Baltimore, Maryland 21203

ABSTRACT

Under contract to the University of Maryland Westinghouse has been investigating the potential for hardware implementation of higher level algorithms associated with the image understanding process. The program is sponsored by DARPA and monitored by the Army's Night Vision and Electro-optical Laboratory. In this report seven current algorithms are defined, and one example (the light/dark, edge-no edge relaxation process, with borderness) is examined in detail, both as regards the computation of initial probabilities, and of the relaxation process itself.

The complexity of emerging algorithms has created a severe processing load for the general-purpose computer. With thirty minute running times for a single image sample, statistical testing becomes prohibitive for a data base of several hundred samples. Westinghouse proposes to attack this problem with the use of array processors. The approach is discussed in this paper. It is noted that software development for array processing may also simplify the hardware implementation effort.

INTRODUCTION

Past effort on this program has been concentrated on the definition of digital architecture for implementing the image processing algorithms developed at the University of Maryland (UMd). Recently, however, a need has emerged to provide support to UMd in the statistical testing of complex algorithms. To understand this need, consider that the processing of a relaxation algorithm for a small image window may require more than 30 minutes of running time on a general-purpose computer. Typical data bases, on the other hand, will contain several hundred image samples. Consequently, the demand on the computer to obtain significant statistical results becomes prohibitive.

There are at least three ways to overcome this dilemma. The most obvious way is to simplify the algorithms. In actual practice, simplification usually does occur. However, the search for new, more powerful appraoches rapidly overcomes the reduction in complexity. It is concluded that means must be found for dealing with very large loads.

A second approach is to build special-purpose hardware capable of adequate throughput rates. The design of such hardware is the subject of our continuing effort. Apart from the obvious drawbacks of time and expense, however, there is a heavy requirement for programmability. The variety of the algorithms which one would like to examine is almost unlimited. The inflexibility of almost any special-purpose hardware could be expected to undesirably restrict the algorithm design and evaluation.

A third approach, and one which we are now investigating, is the use of a fully programmable array processor, with a throughput in excess of 10^8 operations per second. Westinghouse is a leader in array processor development, with particular application to military problems. At present, this is rack-sized equipment. However, the Universal Array development described at the last workshop (Ref. 1) is directed specifically toward the LSI implementation of array processors. Consequently, there is the prospect that a satisfactory performance demonstration on the array processor can be readily translated into a miniaturized form with a minimum of program modification.

The use of the array processor for statistical testing appears to offer the advantages of flexibility and short response time, under the assumption that such a machine can in fact accommodate the UMd algorithms. We are now investigating this matter, as will be discussed below.

The evolution from an image processing concept to miniature hardware might follow the paths shown by Figure 1. After initiating the concept, UMd would develop the algorithm and would perform a feasibility demonstration of its performance on a small set of test samples. Westinghouse would adapt the algorithms for the processing of a statistical test on the array processor, using a test data base approved by NVEOL. In a parallel effort, Westinghouse would investigate the LSI implementation of the algorithms, based on all available logic families. It would also consider their implementation with Universal Array architecture. The results of the statistical test, and the recommendations for implementation would be available for further action by a military agency such as NVEOL.

In the next section we review the prospects for algorithm evaluation using the Programmable Array Processor. This is followed by a discussion of the prospects for implementation of seven algorithms currently under examination at UMd.

IMPLEMENATION OF IMAGE PROCESSING ALGORITHMS ON AN ARRAY PROCESSOR

Westinghouse has developed a family of Programmable Array Processors (PAP) designed for high-speed signal processing. In general, the PAP is a single unit that is specifically structured for high-speed, iterative processing on sets of data commonly referred to as "vectors". If, for example, we consider each line of digitized video data as an input "vector", then image processing of consecutive lines of image data should be accomplished at high speed rates. The PAP is "programmable"; that is, its operation is controlled by a computer. The PAP is "highly programmable", which means that it has complex, higher-order instructions and is capable of autonomous operation and decision making.

The Westinghouse PAP is currently in its third generation of hardware development, with the next generation being designed. Systems applications include high-speed video and radar signal analysis. New designs will make use of Universal Arrays and other LSI circuitry, as well as modular architecture for improved data handling and data throughput rates.

The video input data is stored into bulk memory under program control of an internal general-purpose computer, which includes the capability to reformat and re-order the data. See Figure 2. The Vector-Array Processor (VAP) is a four-channel processor that performs most of the vector arithmetic instrucitons, such as MULTIPLY/ADD, COMPARE, ACCUMULATE, LOGICALS, TABLE LOOK-UP, DOWN SAMPLE, UP SAMPLE, SQUARE, and DETECT. The basic control functions such as loop controls, branching, and indexing are provided in both the VAP and the internal general-purpose computer.

The VAP usually operates two or three orders of magnitude faster than commonly used computers doing iterative processing on data in vector format. The efficient data interface which is capable of video data input rates maintains high data throughput rates for the PAP system. The unit is structured to be very efficient at ordinary arithmetic operations such as multiply and add with fractional scaling, so it is expected to be well suited for complex algorithms such as relaxation.

Westinghouse is investigating the increase in image processing capability that can be achieved by performing the processing functions on the Westinghouse Programmable Array Processor (PAP). It is currently intended that this effort will have several goals, which are principally long-term goals, but with significant results from the initial effort. Specific goals are the following:

1. Analysis of various potentially useful image processing algorithms to obtain a preliminary determination of their compatibility with vector processing format.
2. Estimation of PAP data throughput rates for the principal algorithms. These estimates will include not only the adds and multiplies associated with the repetitive linear operations, but the control overhead and data storage and transmission that is a significant limit on array processing.
3. Evaluation of algorithms for compatibility with the line-at-a-time processing typical of a vector processing architecture. This should include algorithm effectiveness, complexity, and potential for further optimization.
4. Evaluation of compatibility of pipeline processing (typical of the Westinghouse AUTO-Q system) Vector processing (typical of the Westinghouse PAP system) and parallel processing arrays with the basic image processing algorithms. Since there is no known existing hardware for line-at-a-time, parallel processing arrays a potential architecture will be developed for analysis.
5. Comparative analysis of the programmability and potential speed capability of pipeline processors, vector processors and parallel processing arrays, using current fabrication technology and ultimate potential technologies (e.g. CCD's, high speed universal arrays, VHSIC).
6. Depending on the potential capabilities of the parallel processing arrays (determined above) continue development of an architecture and system design for a demonstration unit, and develop the capability for simulating its operation on the PAP.

HARDWARE IMPLEMENTATION

This section describes the algorithms which are currently under evaluation for implementation in hardware. One of these, the "light/dark, edge/no edge" relaxation process, with "borderness", is examined in some detail as representative of the implementation process. First, the initial probabilities are considered. This is followed by the relaxation process itself.

ALGORITHM REVIEW

We briefly review seven algorithms developed by UMd which are being analyzed by Westinghouse for hardware implementation. The first four[2] are segmentation algorithms based on the relaxation approach. The fifth[3] is a spot detector which is designed to select thresholds for segmentation. The last two are higher order algorithms for shape classification and texture classification.

1. "Light/Dark" Relaxation Algorithm

   The Light/Dark Relaxation algorithm initially assigns "light" and "dark" probabilities to image pixels based on their gray levels. These probabilities are then iteratively adjusted at each image point (pixel) based on the probabilities at neighboring points; i.e., light reinforces light and dark reinforces dark. This has the effect of adjusting the probabilities initially assigned to noisy pixels so as to make them more consistent with their surroundings. Eventually, the light probabilities at all points of a light region should become uniformly high and vice versa for the dark probabilities, so that thresholding becomes easy and should produce noise free results. This algorithm and its implementation were described in detail in the Third, Fourth, and Fifth Quarterly Reports on this program.

2. "Edge/No Edge" Relaxation Algorithm

   The Edge/No Edge Relaxation Algorithm is a process in which "edge" and "no edge" probabilities are initially assigned to each image point (or, alternately, to each adjacent pair of points) based on the relative values of the gray level differences in various directions around the point. These probabilities are then adjusted based on the probabilities at neighboring points.: no-edge reinforces no-edge; edge reinforces edge if they smoothly continue one another, and reinforces no edge (and vice versa) if they are alongside one another. This has the effect of strengthening the appropriate edge probabilities at points that lie along smooth edges, and strengthening the no edge probability elsewhere, so that edge detection should yield less noisy results. This algorithm and its implementation were described in detail in the Fourth and Fifth Quarterly Report.

3. Joint "Light/Dark, Edge/No Edge" Relaxation

   This algorithm combines the above algorithms and allows them to interact in much the same manner as the individual probabilities interact in either of the above algorithms. The interaction between light/dark and edge/no edge takes place at the relaxation level which substantially increases the computational load.

4. Joint "Light/Dark, Edge'No Edge" Relaxation with "Borderness"

   This algorithm produces results similar to the Joint "Light/Dark, Edge/No Edge" Relaxation Algorithm described above; it allows interaction between the "Light/Dark" and "Edge/No Edge" Algorithms using a Borderness concept. This essentially, provides interaction at the initial probability level rather than the relaxation level, and eases the computational burden substantially. The relaxation computation is done in the Light/Dark mode. Implementation of this algorithm is a combination and extension of the relaxation work performed previously. Here, the probability of light is initially high only adjacent to edges on their light sides. This gives an array of borderness values which are high on the light size of edges and and low elsewhere.

5. Spot Detector

This algorithm permits each target within a image to be thresholded individually. Lower resolution renditions of the original image are obtained by 4x4 averaging; successive averaging will produce a spot (one pixel) target. A biased 3x3 LaPlacian Operator is scanned across the lower resolution image and responses are analyzed for frequency and spatial proximity. Given a positive response, a threshold is selected by averaging the eight background pixels surrounding the target, and averaging this quantity with the target gray level. The result is used to threshold the target in the original, high resolution image.

6. Shape Classification by Relaxation[4,5,6]

The purpose of this algorithm is to recognize shapes described by closed boundary curves of more complex shapes such as airplanes. Basically they are segmented into a number of parts and sequences (triples) of these parts are examined for consistency using probabilities relaxation. The segmented parts could be nose, right wing, left wing, and tail. Since the segmentation is imperfect, initial probabilities are assigned, analytically, to the various segments. Then sequences of adjacent segmented parts are used to reinforce the identification of particular segments.

7. Texture Primitive Extraction[7,8]

Many textures can be characterized as a collection of primitive elements, i.e. connected regions satisfying certain properties. Here, the primitive extraction is done using edge-based techniques.

IMPLEMENTATION OF "LIGHT/DARK EDGE/NO EDGE" with "BORDERNESS" ALGORITHM

We describe the implementation of the joint "Light/Dark, Edge/No Edge" Relaxation with "Borderness" Algorithm because it represents a summary of the work performed on relaxation and its implementation would be capable of landling some of the higher order relaxation algorithms such as the Shape Classification Algorithm. We shall concentrate first on the initial probability computations.

INITIAL PROBABILITIES (Edge/No Edge)

The Edge/No Edge Algorithm finds the largest edge value over a 3x3 neighborhood for each of eight directions by multiplying the gray levels in the 3x3 array by a mask rotated through eight positions. The 3x3 array is shown in Figure 3 together with the eight mask position.

The edge value for first mask position is

$$\begin{aligned} e(1) = {} & A(-1) + B(0) + (C(+1) \\ & +D(-1) + E(0) + F(+1) \\ & +G(-1) + H(0) + (I(+1), \end{aligned}$$

and the edge value for a particular pixel is $e = \max_i e(i)$, $i = 1,2,\ldots 8$. We note the symmetry in the edge computations shown in Figure 4a. In Figure 4b, we rearrange the interiors of the quantities such that the middle pixel in each quantity is also the middle pixel, geometrically, between its neighbors in the 3x3 array. Then one clockwise shift around the outer eight pixels followed by a counter clockwise shift forms the quantities shown in Figure 5. Then with four additional shifts in either direction, the quantities of Figure 4a appear in the outer eight pixel positions as shown in Figure 6. The number of operations to form these quantities are six shifts and three adds performed in parallel over eight units. If we assume that each pixel is represented by a processor capable of 20 million operations/sec. then the nine operations consume 450 nanoseconds. Hence, five sets of eight processors are capable of performing the edge computations at video rates. We delete the $e = \max_i e(i)$ step because it is not included in the borderness computation.

INITIAL PROBABILITIES (EDGE/NO EDGE WITH BORDERNESS)

In computing "Borderness" we add the e(i) value, corresponding to the +1 position of the mask, with the gray level at that pixel position. The resultant quantity is placed at that pixel position. In terms of the mask positions and e(i) quantities, the final borderness mask is shown in Figure 7. With the completion of the edge/no edge computation described in the previous paragraph, the edge values, e(i), are in positions within the computational array shown in Figure 8a. By shifting clockwise four positions, e(1) is in the middle of the

+1
+1
+1 column for mask 1. Similarly, each of the other edge values are in the middle position of their +1 sequence. Then by shifting one position clockwise, followed by a counter clockwise shift. (Figure 8c), the appropriate triplets in Figure 7 can be formed. Thus to form the Borderness values, six shifts and two adds are required for a total of 400 nanoseconds.

The total time to compute Borderness is 950 nanoseconds which requires 10 sets of 8 processors for real time operation.

INITIAL PROBABILITIES (LIGHT/DARK)

We define $p_{xy}(\lambda) = \frac{gl_{xy} - gl_{min}}{gl_r}$ as the probability of a pixel at position xy being dark where gl_{xy} is the gray level at pixel xy, gl_r is the gray level range over the image, gl_{min} is the minimum gray level over the image, and λ refers to dark. Then an estimate of the probability of any pixel in the image being dark is $\overline{p}(\lambda) = \frac{1}{n} \sum_{xy} p_{xy}(\lambda)$ where n is the number of pixels in the image. An estimate of the joint probability of the center (ith) pixel (of a 3x3 window) being dark and its eight neighbors (jth pixel) being light is $p_{ij}(\lambda\lambda') = \frac{1}{n} \sum_{xy} p_{xy}(\lambda) p_{x+i,y+j}(\lambda')$ where λ' refers to light. Note that $p_{xy}(\lambda') = 1 - p_{xy}(\lambda)$, i.e. the probability of light is one minus the probability of dark. There are four joint probabilities $p_{ij}(\lambda\lambda)$, $p_{ij}(\lambda'\lambda)$, $p_{ij}(\lambda'\lambda')$, and $p_{ij}(\lambda\lambda')$ to be computed for each of the eight neighbors for each pixel in the image. By substitution, assuming $gl_{min} = 0$, we obtain

$$p_{ij}(\lambda'\lambda') = \frac{1}{n} \frac{1}{gl_r^2} [1 - gl_r \sum gl_{xy} - gl_r \sum (gl_{x+i,y+j}) + \sum gl_{xy}\, gl_{x+i,y+j}],$$

and, in general, any of the joint probability expressions can be written as

$$p_{ij}(\ ) = f[\sum gl_{xy}, \sum gl_{xy}\, gl_{x+i,y+j}, \sum gl_{x+i,y+j}].$$

Further, it is shown in the Fifth Quarterly Report that $\sum gl_{x+i,y+j} = h(\sum gl_{xy}, \sum r_k, \sum c_l)$. That is, the sum over the neighboring pixels can be found from the sum over all the pixels minus the sum over two rows (from $1, 2, r_{k-1}, r_k$) and two columns (from

$1,2,c_{l-1},c_l)$ where k is the total number of rows and l is the total number of columns. This allows the computation of $\Sigma\, gl_{xy}$, $\Sigma\, gl_{x+i,y+j}$ as an accumulation plus two arithmetic operations. For the $gl_{x+i,y+j}\, gl_{xy}$ product, we note the symmetry properly. That is, let us change notation such that i,j are running indices over the entire frame. Then the center pixel to lower right pixel combination instead of being $gl_{xy}\, gl_{x+i,y+j}$ becomes $gl_{ij}\, gl_{i+1,j+1}$. Clearly, $gl_{ij}\, gl_{i+1,j+1}$ is equal to $gl_{i+j,j+1}\, gl_{ij}$ which is the center to upper left combination in the next row and one column to the right. With this reciprocity in mind, we need only compute four of the $gl_{ij}\, gl_{i+1,j+1}(gl_{xy},\, gl_{x+i,y+j})$ combinations directly with the operator shown in Figure 9. Assuming a 2's complement addition in eight steps for an 8 bit by 8 bit multiplication, there are 400 nanoseconds per multiplication and 1600 nanoseconds per pixel. Hence, 16 processors are needed (assuming a 50 nanosecond clock cycle) to produce the initial probability products for light/dark relaxation in real time, i.e. video rates of 30 frames per second.

In summary, 96 processors (20 mega operations/sec) are required to produce the initial probability computations for the Joint "Light/Dark, Edge/No Edge" with "Borderness" Relaxation Algorithm at 30 frames per second.

THE RELAXATION COMPUTATION

In the First and Second Quarterly Reports, we described a number of commercially available bit-sliced architectures, e.g. AMD 2901, AMD2903, MC 10800, etc. in terms of their instruction repetoire, speed, power, and size. We concluded that, although bit-sliced architecture offered substantial advantages over conventional microprocessors, the size and speed were not adequate for airborne image processing applications, particularly missiles. We then concentrated our efforts on the use of Westinghouse Universal Arrays in the Third and Fourth Quarterly Reports, as a tool for implementation. We described the relaxation calculations in terms of Systolic Arrays, composed of Universal Array multipliers. We continue that effort here, and broaden it, with the addition of a programmable Westinghouse Universal Array (ALU) Arithmetic Logic Unit. It is assumed that the ALU is 8 bits wide, has an instruction set comparable to the AMD 2903, performs 20 million arithmetic operations per second, is contained on a .5 inch x .5 inch chip and performs multiplication on-chip using 2's complement addition. A 8 bit x 8 bit multiplication takes 400 nanoseconds. In the previous section, this processor was applied to the initial probability computations for the Joint "Light/Dark, Edge/No Edge" Relaxation Algorithm with "Borderness". In this section, the processor is applied to the relaxation computation for that algorithm.

To implement the relaxation computation, assume that the quantities $p_{ij}(\lambda\lambda')$, $r_{ij}(\lambda\lambda')$, $p_i(\lambda)$, $p_j(\lambda')$ and $\overline{p}(\lambda)$ have been computed as part of the initial probability calculations.

The first quantity computed is

$q_{ij}^{k+1}(\lambda) = \sum_{\lambda'} p_i^k(\lambda)\, p_j(\lambda')\, r_{ij}(\lambda\lambda')$. For $\lambda' =$ 1,2 the quantity becomes

$$q_{ij}^{k+1}(\lambda) = p_i^k(\lambda)\, p_j^k(1)\, r_{ij}(\lambda,1) + p_i^k(\lambda)\, p_j^k(2)\, r_{ij}(\lambda,2).$$

For each possible value of λ, the above quantity then becomes:

$$q_{ij}^{k+1}(1) = p_i^k(1)\ p_j(1)\ r_{ij}(1,1) + p_i^k(1)\ p_j^k(2)\ r_{ij}(1,2) \qquad (1)$$

$$q_{ij}(2) = p_i^k(2)\ p_j(1)\ r_{ij}(2,1) + p_i^k(2)\ p_j^k(2)\ r_{ij}(2,2), \qquad (2)$$

a pair of expressions. The $p_{ij}^{k+1}(\lambda) = q_{ij}^{k+1}(\lambda)/$ $q_{ij}^{k+1}(\lambda)/\Sigma_{\lambda'}\ q_i^{k+1}\ (\lambda')$ expression is then one of the above pair in the numerator over their sum in the denominator. So the computation of (1) and (2) is the basic computation of relaxation and other expressions $p_{ij}^{k+1}(\lambda)$, $p_i^{k+1}(\lambda)$ are mere manipulations of this pair. Consider, the computation of expression (1) and (2). The i index refers to the center pixel in the 3x3 array; the index j refers to each of the eight neighbors. But as discussed earlier, we can use the same operator, shown in Figure 10, to compute $q_{ij}^{k+1}(\lambda)$. In other words, at each center pixel, we need only compute the $q_{ij}^{k+1}(\lambda)$ appropriate for four of the eight neighbors, numbered arbitrarily in Figure 10. The computations required are:

$$q_{c4}^{k+1}(1) = p_c^k(1)\ p_4^k(1)\ r_{c4}(1,1) + p_c^k(1) \cdot p_4^k(2)\ r_{c4}(1,2)$$

$$q_{c4}^{k+1}(2) = p_c^k(2)\ p_4^k(1)\ r_{c4}(2,1) + p_c^k(2)\ p_4^k(2)\ r_{c4}(2,2)$$

$$q_{c6}^{k+1}(1) = p_c^k(1)\ p_6^k(1)\ r_{c6}(1,1) + p_c^k(1)\ p_6^k(2)\ r_{c6}(1,2)$$

$$\vdots$$

$$q_{c8}^{k+1}(2) - p_c^k(2)\ p_8^k(1)\ r_{c8}(2,1) + p_c^k(2)\ p_8^k(2)\ r_{c8}(2,2)$$

There are eight expressions of the form:

$$a\ b\ c + a\ d\ e = a\ (bc + de),$$

that is, multiply, multiply, add, and multiply. If we assume 400 nanoseconds for an on chip multiply, and 50 nanoseconds for an add, the computation time is 1250 nanoseconds for one $q_{ij}^{h+1}(\lambda)$ quantity and 8x1250 = 10,000 nanoseconds for 8.

If we assume an image frame containing 330,000 pixels (standard 525 line TV), then processing at video rates (30 frames/sec.) means a processing rate of 10 million pixels/sec. or an allowable time interval between pixels of 100 nanoseconds. With two processors, the allowable time between pixels is 200 nanosecond for each processor; with twenty processors, the allowable time per processor is 2000 nanoseconds; and with 100 processors, the allowable time per processor is 10,000 nanoseconds. Hence, 100 ALU's of the type just described produces the allowable time between pixels or a processing rate of 2,000 mega operations (2 billion)/second. Assuming 0.5 inch x 0.5 inch for each ALU chip, a 3 inch x 3 inch space accommodates 36 ALU's and a board pair contains 72. With 1/2 inch centers between board pairs, a 3 inch x 3 inch x 3 inch volume contains six board pairs or 12 boards. Only three of these boards contain the 100 ALU's necessary for the $q_{ij}^{k}(\lambda)$ computation. This means that the frame is divided into 100 vertical strips and each processor is responsible for approximately 50 pixels in the image. The preliminary design allows the relaxation iterations to be computed at frame rates. Thus, if five iterations are necessary for convergence, the actual cueing rate is six frames per second.

To reduce computation time by a factor of 5, we could utilize 500 processors, but this would exceed the desired volume. More simplification are necessary in the computations. In summary, preliminary analysis indicates that it may be possible to perform relaxation computations at a rate of six frame/sec. in a volume of 3 inches x 3 inches x 3 inches using 100 programmable processors.

With regard to hardware implementation, the following conclusions have been reached:

1. Joint LIght/Dark Edge/No Edge with Borderness which incorporates interaction between Light/Dark Edge/No Edge at the initial probability level is easier to implement than the Joint Light/Dark Edge/No Edge, results appear to be comparable.
2. This algorithm can be accommodated, according to preliminary analysis, in a 3x3x3 inch volume at six frames per sec. assuming five iterations for convergence.
3. The spot detection algorithm appears to produce thresholds in a relatively simple manner, can be accommodated in a 3x3x3 inch volume and has application in tactical missiles.
4. Shape Classification by Relaxation can be accommodated by the 100 ALU array processor found in the 3x3x3 volume with some dynamic reconfiguration of architecture.
5. Texture analysis offers an opportunity to attack the problem of forming edge based groups or primitives using memory array techniques which are now more attractive based on memory advances and the 100 ALU array.

REFERENCES

1. Relaxation, Systolic Arrays, Willett, T.J., Brooks, C.W., Tisdale, G.E., Proceedings Image Understanding Workshop, page 164 DARPA, April, 1979.
2. Image Understanding Project Status Report, Rosenfeld, A., Proceedings: Image Understanding Workshop, page 14, DARPA, April, 1979.
3. Unpublished work under DARPA-NV&EOL Sponsorship, Schaer, M., Rosenfeld, A. to be published in fall, 1979.
4. Breaking Substitution Ciphers Using a Relaxation Algorithm, Peleg, S., Rosenfeld, A., TR-721, Computer Science Center, University of Maryland, January, 1979.
5. Some Relaxation Experiments Using Triples of Pixels, Eklundh, J., Rosenfeld, A., TR-752, Computer Science Center, University of Maryland, April, 1979.
6. Shape Segmentation Using Relaxation, Rutkowski, W.S., Peleg, S., Rosenfeld, A., TR-762, Computer Science Center, University of Maryland, May 1979.
7. Texture Primitive Extraction Using an Edge-Based Approach, Hone, T., Dyer, C., Rosenfeld, A., TR-763, Computer Science Center, University of Maryland, May 1979.
8. Second Order Statistics of Texture Primitives, Wang, S. Wu, A., Rosenfeld, A., TR 779, Computer Science Center, University of Maryland, July, 1979.

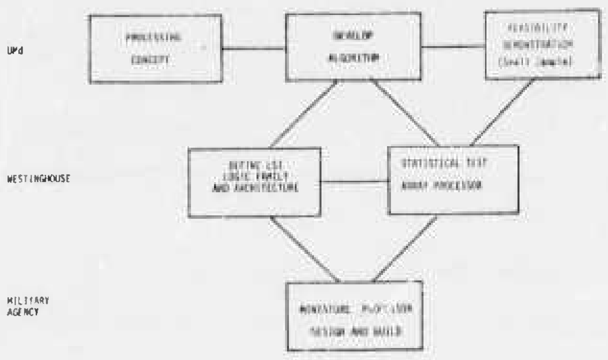

UMd
WESTINGHOUSE
MILITARY AGENCY
PROCESSING CONCEPT
DEVELOP ALGORITHM
FEASIBILITY DEMONSTRATION (Small Sample)
DEFINE LSI LOGIC FAMILY AND ARCHITECTURE
STATISTICAL TEST ARRAY PROCESSOR
MINIATURE PROCESSOR DESIGN AND BUILD

Figure.1. **Evolution from Processing Concept to Miniature Hardware**

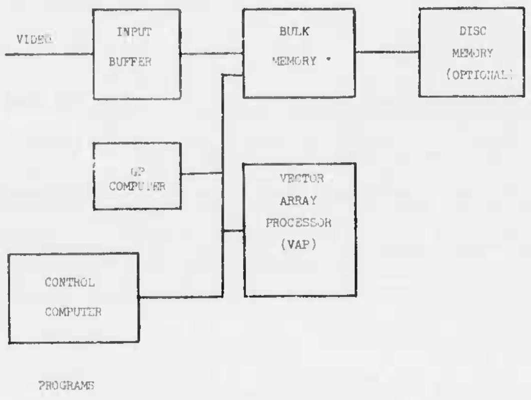


Figure 2. Programmable Array Processor (PAP) Block Diagram

```
                    A  B  C

                    D  E  F

                    G  H  I

-1 0 +1   0 +1 +1   +1 +1 +1   +1 +1  0   +1  0 -1
-1 0 +1  -1  0 +1    0  0  0   +1  0 -1   +1  0 -1  . . . . .
-1 0 +1  -1 -1  0   -1 -1 -1    0 -1 -1   +1  0 -1

   1.        2.         3.         4.         5.
```

Figure 3. 3x3 Neighborhood and Successive Masks

```
Mask 1. +(C+F+I) - (A+G+D)
     2. +(B+C+F) - (H+G+D)
     3. +(A_D_C) - (I+H+G)
     4. +(D+A+B) - (F+I+H)
  ------------------------
     5. +(A+D+G) - (C+F+I)
     6. +(H+D+G) - (B+C+F)
     7. +(I+H+G) - (A+B+C)
     8. +(F+I+H) - (A+B+D)
```

Figure 4a. Mask Tabulations

(DAB)	(ADG)
(ABC)	(HDG)
(BCF)	(IHG)
(CFI)	(FIH)

Figure 4b. Interiors Rearranged

D+A+B	A+B+C	B+C+F
A+D+G		C+F+I
D+G+H	G+H+I	H+I+F

Figure 5. Formed Quantities

D+A+B H+I+F	A+B+C G+H+I	B+C+F D+G+H
A+D+G C+F+I		A+D+G C+F+I
D+G+I B+C+F	G+H+I A+B+C	H+I+F D+A+B

Figure 6. Four Successive Shifts

[e(4)+e(3)+e(5)]	[e(3)+e(2)+e(4)]	[e(2)+e(1)+e(3)]
[e(5)+e(9)+e(6)]		[e(1)+e(2)+e(8)]
[e(6)+e(7)+e(5)]	[e(7)+e(6)+e(8)]	[e(8)+e(1)+e(7)]

Figure 7. Borderness Mask

e(2)	e(3)	e(4)
e(1)		e(5)
e(8)	e(7)	e(8)

Figure 8a. Initial Position

e(4)	e(3)	e(2)
e(5)		e(1)
e(6)	e(7)	e(8)

Figure 8b. Final Position

$$\begin{tikzcd} \cdot \quad \cdot \quad \cdot & e(1) \\ \cdot \quad \cdot \quad \cdot & e(1) \arrow[u] \arrow[d] \\ \cdot \quad \cdot \quad \cdot & e(1) \end{tikzcd}$$

Figure 8c. Bi-directional Shift

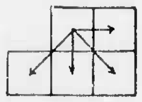


Figure 9. Reciprocity Operator

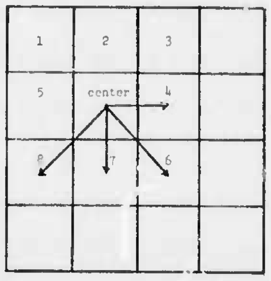


Figure 10. Array of Pixels

ZMOB: A Mob of 256 Cooperative Z80A-Based Microcomputers

Chuck Rieger

Computer Science Department
University of Maryland
College Park, MD 20742

1. INTRODUCTION

Current directions of computer science and computing in general are toward more parallel machine architectures and distributed models of computing based upon these new architectures. Broadly speaking, parallel architectures fall into two categories: parallel synchronous machines (typified by ILLIAC IV), which execute the same code synchronously in many processors operating on different data, and parallel autonomous machines (typified by CM*), in which many independent processors can be put to work on different aspects of a large computation.

In past and much current work, emphasis has been on the former variety of machine [1-3]. Recently, however, there has been considerable interest in highly parallel architectures capable of supporting complex distributed computation via a large number of autonomous processors [4,5]. While many interesting machines have been proposed or are currently being developed, there has apparently been no specific attempt to build a machine with a truly large number of autonomous processors, each having substantial independent computing power.

ZMOB is such a machine, currently under design and simulation at Maryland. Architecturally, ZMOB is a collection of 256 identical but autonomous Z80A-based microcomputers (processors). Each processor (Fig. 1) comprises 32K bytes of 375 ns read/write central memory (expandable to 48K bytes), up to 4K bytes of resident operating system on 450ns EPROM, an 8-bit hardware multiplier, and interface logic for communications functions. (This is a non-trivial microcomputer, comparable in size and power to the average small business or personal computer.) Two processors will share each PC board, making a total of 128 processor boards mounted in several rack cabinets, and supplied by a rather hefty power supply. Although the machine will initially consist of 256 processors, its architecture is extensible (in principle) to any number of processors. In practice, we will anticipate extensibility to 1024 processors. The current cost estimate for the 256 processor machine is $100K.

Good intercommunication pathways and bandwidths are critical to the success of any highly parallel machine. As described below, we have what we feel is a very attractive solution to inter-processor communication, and processor-to-outside-world communication. The strategy will be non-blocking (e.g., there can be 128 full-speed conversations between pairs of processors), and will give each processor the illusion of data communication rates as high as the Z80A itself can manage. In one mode, for example, data rates into or out of ZMOB can exceed 20 megabytes per second.

2. BACKGROUND AND MOTIVATION

The ZMOB idea sprang originally from needs of the Computer Vision Lab at Maryland. In certain vision tasks based on relaxation algorithms [6], each pixel of, say, a 512 by 512 image must be processed once per "iteration". The processing of each of the quarter million pixels is identical, and independent of the processing of other points during one iteration. A typical relaxation algorithm will require 5-10 iterations to converge. The particular algorithm used will vary with the application.

In a complex relaxation algorithm, each pixel is represented by a "probability vector" of perhaps 10 bytes, and the per-point, per-iteration computation in such an algorithm might involve 1000 8-bit integer multiples and a corresponding number of additions. Rough calculations show that one complete iteration can therefore require upwards of 250,000,000 multiplies and a comparable number of adds, a staggering computation which requires hours on a medium-size conventional machine. Our preliminary studies (relatively complete Z80A code, hand-simulated timing results) indicate that ZMOB will require on the order of 100 seconds for such a computation.

Although motivated by relaxation processing needs, it quickly became obvious that a machine with such a great computing potential ought to be general-purpose as a research tool for distributed computing models in all of computer science. Specifically, it became of concern that the geometry of intercommunication paths not be unduly biased by the machine's applications in vision (where 4-neighbor adjacency is natural), and that the 8 megabytes of high-speed central memory not be inaccessible to computations desiring to view the machine as a large single address space. The design of the intercommunication system reflects these concerns, and results in a machine which is both general-purpose, and which (from timing studies) supports the initial vision applications as fast as any other special-purpose Z80A-based archictecture could.

3. BASIC PROCESSOR ARCHITECTURE

ZMOB is a collection of autonomous Z80A-based microcomputers. The Z80A is an 8-bit microprocessor chip with a 158 instruction repertoire, two sets of 7 8-bit registers, and several 16-bit registers for stack pointer, program counter, and indexed addressing. It is a stack machine with a 64K byte address space, 256 logical I/O ports, and a T-cycle time of 250ns. A typical instruction will require about 2 microseconds to execute, and there are several rather powerful block search and transfer instructions. High-speed vectored interrupt linkage is another virtue of the chip, which sells in quantities for less that $15.

In the initial conception, the plan was to assign each ZMOB processor to two scan lines of image data in a 512 by 512 pixel image. In such a machine, each processor would be connected to its two adjacent processors (handling adjacent scan lines), and to an external controlling computer (e.g., a PDP-11), all over interrupt driven 8-bit parallel ports with handshaking. At power-on, ZMOB would be cleared, bringing each processor back to its basic resident operating system. This system would allow for the loading of applications programs and parameters into the processor's RAM. After initialization, all processors would be forced into their DMA condition while the external machine, having access to the individual processors' address spaces (as pieces of one large virtual space), loaded in the starting image data. After loading and removal of the DMA condition, the external machine would broadcast a system-wide start command over all control ports. Once running, each processor would request information from its neighboring processors, compute two pixels' worth of probability vector updates, then advance to the next of the 512 pixels across its two scan lines. The operation would progress (pretty much synchronously) in all processors simultaneously until, at the end of the scan line, each would broadcast "ready" messages to the external computer. When all had been accounted for, the external computer would again force all processors to their DMA state, read the iteration's results for TV display update, then release the processors on the next iteration.

Timing simulations showed that such a machine would be quite profitable. For example, the time required to compute each image pixel's 3 by 3 8-bit gray-scale average in a 512 by 512 image would be about 2 fifths of a second, while the time required to run a simple edge detector over a 512 by 512 image would be about one second. Even an elaborate relaxation algorithm involving 10 labels per image point would complete each iteration in about 100 seconds, orders of magnitude faster than presently possible on a conventional machine.

The initial conception of the machine rapidly evolved into a design capable of supporting a variety of distributed computing models, in addition to the vision models from which the idea came. The current design, ZMOB, supports the original vision applications within the same time estimates, and will result in a machine that is a flexible and general purpose research tool.

4. THE CONVEYOR BELT

In the current design, there are no direct neighbor-neighbor communication paths. Rather, each processor is a mail stop on a high-speed, synchronous "conveyor belt" (Fig. 2). The ZMOB portion of the conveyor belt is thus 256 positions long (but indefinitely extendible), and resembles certain existing synchronous ring networks in its concept [7] (although the types of messages passed are quite different). The external controlling computer, and perhaps other devices such as high speed disk interfaces, are additional mail stops on the conveyor belt.

Each mail stop (Fig. 3) is associated with a processor, and is physically a part of that processor's PC board. Mail stops are connected together over dedicated backplane bus lines. Each mail stop is a high-speed synchronous latch capable of switching data between the processor and the conveyor belt. While the optimal width of the conveyor belt has not been determined, we are presently conducting timing studies based on a width of four fields of between 8 and 10 bits each: source ID, destination ID, data, and control.

Conceptually, the conveyor belt's role is to accept a message from a processor and deliver it to another directly or indirectly referenced processor, or population of processors. Ideally, we would like the conveyor belt to serve as a non-blocking message switcher, i.e., one in which n/2 simultaneous processor-processor conversations could be in progress at maximum Z80A rates. This would give each processor the illusion of having instant access to any other processor.

As it turns out, this ideal is achievable. Aside from DMA, the Z80A's highest memory or I/O data transfer rate is one byte per 5.25 microseconds (achieved during several of the block memory-move instructions). This is a hardware limitation of the Z80A, and cannot be improved upon by clever programming techniques. To act as a non-blocking message switcher, the conveyer belt needs only to make one complete revolution every

5.25 microseconds so that a specific position on the belt (a bin) will always be available for each processor's next memory transfer every 5.25 microseconds. Conventional high-speed digital electronics can support the approximately 50 mhz shift frequency required for such a conveyor belt. While engineering and economic considerations might dictate a somewhat slower conveyor belt in the 20 mhz range, even at a lower rate, most forms of interprocessor communication can proceed at full Z80A rates.

The conveyor belt is synchronized by two system-wide control lines, the conveyor belt shift clock, and the index pulse. The shift clock controls the basic movement of data into and out of each mail stop, and hence around the conveyor belt. The index pulse is emitted once per complete revolution of the belt, and signifies to each processor that its own bin is under the processor.

Each processor owns the bin indicated by the index pulse. When this bin is at the processor, any data waiting in the STAGING REGISTER will be taken onto the conveyor belt. The staging register, loaded byte-wise by the Z80A at its convenience, and armed when its data field has been loaded, serves to synchronize the otherwise asynchronous operations of the processor and conveyor belt. Outbound data will only be accepted when the processor's bin is flagged as empty by a bit in the control byte (i.e., if the message is not consumed by the intended destination, it will be retained on the belt, unless the originating processor has indicated that it wishes to intercept its own transmission if not consumed in one revolution).

Outbound data requires only a go, no-go decision about whether the bin is free to accept the contents of the staging register. For the inbound pathway, each mail stop requires a small amount of high-speed decision logic for intercepting conveyor belt messages directed at its processor. In addition to its numerical address on the conveyor belt, each processor can advertise a category code. When armed, this category code (any combination of zeroes, ones, and don't-care's) will accept any message whose destination field matches the code, permitting call-by-capability in addition to call-by-name.

When deemed appropriate, a conveyor belt message is lifted off the belt into the processor's HOLDING REGISTER, and the bin from which it came marked as empty. It is appropriate to lift a message into the holding register only when the holding register is empty and the inbound decision logic determines its processor to be an appropriate receiver of a message if (1) the message is directed to the processor by direct numerical address, (2) the capability code of the message matches the processor's capability code, or (3) the processor's own message has arrived back at the processor after one complete revolution on the belt without being read. Each of these three receive conditions can be selected or deselected by the processor via processor-writable control bits in the inbound decision logic. When none are enabled, the mail stop will accept no conventional messages.

In addition to these three receive conditions, there is a fourth condition used in conjunction with high-speed block bursts between a pair of processors. In this mode, neither processor of the pair will be inspecting or setting any conveyor belt field but the data field. The receiver must therefore be in a mode which exludes all inbound messages other than those originating from the processor with which it is communicating. In this private conversation mode, a fourth, overriding component of the inbound decision logic permits the receiver to identify an exclusive source of inbound messages. When in this mode, only a message whose source ID matches the contents of the EXCLUSIVE SOURCE register will be intercepted by the inbound decision logic.

When a conventional message is accepted into the holding register by the inbound decision logic, the BELT DATA AVAILABLE status flag is set, and a maskable interrupt generated. The processor can then inspect the message at its convenience by reading the holding register contents as a group of input ports. For block burst mode, in which both the sender and receiver are executing block instructions (and have disabled their interrupts), the inbound decision logic will also control the PROCESSOR WAIT line to provide hardware synchronization between the processor and belt.

Inbound messages can be read either destructively (i.e., consumed) or non-destructively (i.e., noted) by the mail stop, according to another control bit associated with the message. Destructive reads are used for one-one conversations, while non-destructive reads are used for broadcasting messages to the population at large.

For absolute external control, there is a class of conveyor belt control messages that will be unconditionally accepted by a mail stop. Some of these can be directed at a specific processor or class of processors, while others can be broadcast to the population at large (i.e., are not consumed by mail stops, but instead passed along). All control messages release any processor-wait condition, and generate a non-maskable processor interrupt to bring the processor back to its operating system. In this way, the controlling computer maintains ultimate control over ZMOB.

5. PATTERNS OF USE

ZMOB will be a general purpose research tool for distributed and autonomous, parallel computating. As mentioned, each processor itself would be powerful enough to run its own operating system with text editors and high level languages, if it were a stand-alone personal computer attached to a floppy disk system. (For example, with a 48K memory, each processor would be capable of supporting a PASCAL compiler.)

In the vision relaxation applications for which the machine is to be used initially, each processor will be running the same code on its own subregion of the 512 by 512 pixel image. In a large relaxation algorithm, each pixel will be represented by, say, a 10 byte probability vector, meaning that each processor will work on 10,240 bytes of image data. This data, together with the relaxation algorithm's code and constant data, will be shipped to each processor at very high speeds over the conveyor belt during initialization. The code and constant data can be loaded at the 5.25 microsecond rate of the Z80A by having the external computer load one byte of data onto the belt in nondestructive read mode each 5.25 microsecond. All processors will note and store each byte via their high-speed block input instruction loops. This means, for example, that all ZMOB processors could accept a 10,000 byte program in just over one tenth second, assuming a conveyor belt speed of 20 mhz. After loading the program, the external computer loads the image data at whatever rate it is able. In this mode, if capable of the high data rate, the external computer could load each revolution of the conveyor belt with the next 256 bytes of image data, each directed to a different processor. Assuming the conveyor belt runs at 20 mhz, and that the external computer is capable of meeting this data rate, the 2.5 million bytes of a 512 by 512 pixel image of 10-byte-deep probability vectors can also be loaded in slightly over one-tenth second. After processing, delivery of results would occur at a comparable data rate.

The vision applications can be characterized as highly parallel, nearly synchronous computations, not dissimilar to those for which ILLIAC IV was designed. However, these applications use ZMOB in a highly structured way. Another obvious mode of operation is one in which each processor runs its own expert code, and the machine is used more as a population of experts in the MICROPLANNER [8], CONNIVER [9], or ACTORS [10] paradigm. We expect much interesting research to open up in this area.

Another mode of operation would segment ZMOB into fiefdoms. Each fiefdom would be a cluster of processors, governed by one agreed-upon distinguished member. This member would be responsible for one ongoing computation, and would use his serfs primarily as extended memory which he could page in as needed. Since the conveyor belt has been designed to be responsive to high-speed data bursts among processors, we will be able to develop a very fast paging system capable of paging rates equal to or surpassing good disks (i.e., no seek latency, but somewhat slower data rates). In this pattern of use, for example, we might run LISP on ZMOB by creating a fiefdom for the evaluator, one for the garbage collector, one for the scanner, one for a real-time debugger/monitor, and so on.

In a more conventional pattern of use, only one of ZMOB's processors would be distinguished as the central computing processor, and all other processors would support high speed paging and memory management for that one processor. This would make all 8 or 12 magabytes of high speed memory directly accessible, and would resemble a large conventional computer with very fast paging potentials. However, since a single Z80A is admittedly not a high throughput machine, it will probably turn out that ZMOB will seldom be used in this mode.

6. CONCLUSION

It is anticipated that ZMOB development will be in full swing by late spring 1980. Before that time, we hope to have a prototype system of 4 to 8 processors running on a small conveyor belt. The project will be supported by several faculty and graduate students, and will hopefully be in relatively complete form by spring 1981. During development, extensive software development tools (assemblers, higher level system languages, simulators, debuggers) will emerge. Hopefully, the machine will also simulate and make possible new theories of distributed problem solving and parallel computing.

ACKNOWLEDGMENTS

Azriel Rosenfeld supplied the vision application which stimulated initial thinking about ZMOB. Through discussions, John Bane, Randy Trigg and Milt Grinberg have greatly contributed to the ideas described here.

REFERENCES

1. Duff, M. J. B., and Watson, D. M., The Cellular Logic Array Processor, Computer J. 20, 1977, 61-72.

2. Slotnick, D. L., Research in the Application and Design Refinement of the Massively Parallel Processing Computer (MPP), Quarterly Progress Reports, University of Illinois, Urbana, IL, Oct. 1977 ff.

3. STARAN, in Enslow, P. H., Jr., ed., Multiprocessors and Parallel Processing, Wiley, NY, 1974.

4. Fuller, S. H., et al., A Collection of Papers on CM*: A Multi-Microprocessor Computer System, Carnegie-Mellon Computer Science Memo, 1977.

5. Dennis, J. B., Misunas, D. P., and Leung, C. K., A Highly Parallel Processor Using a Data Flow Machine Language, CSG Memo 134, Lab. for CS, MIT, 1977.

6. Rosenfeld, A., Iterative Methods in Image Analysis, Pattern Recog. 10, 1978, 181-187.

7. Prime Computer, Inc., Prime Net Reference Manuals.

8. Sussman, G., Winograd, T., and Charniak, E., Microplanner Reference Manual, MIT AI Memo 203a, 1971.

9. McDermott, D., and Sussman, G., The Conniver Reference Manual, MIT AI Memo 259a, 1972.

10. Hewitt, C., Viewing Control Structures as Patterns of Passing Messages, MIT AI Memo 410, 1976.

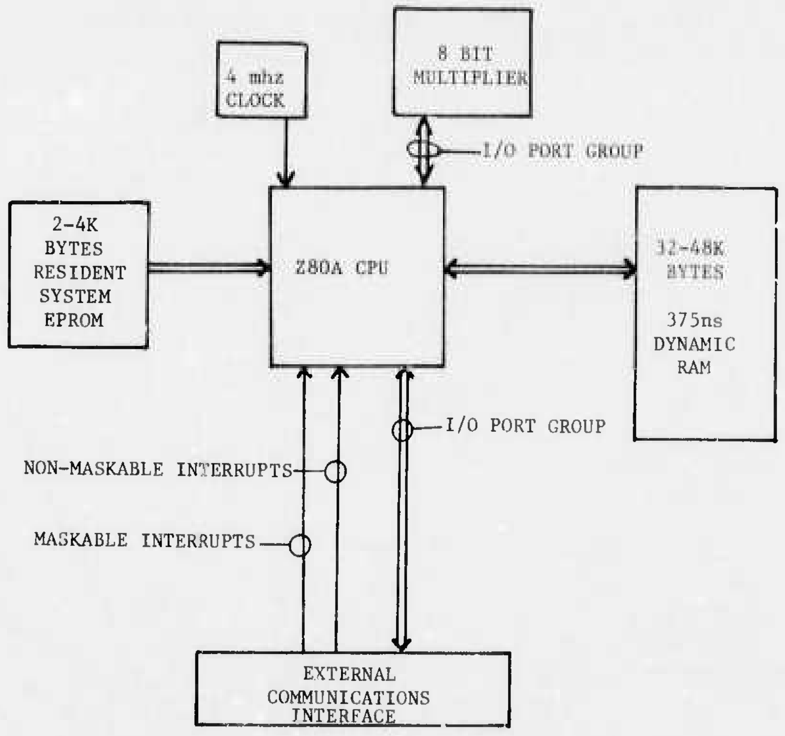


Figure 1. Processor Organization.

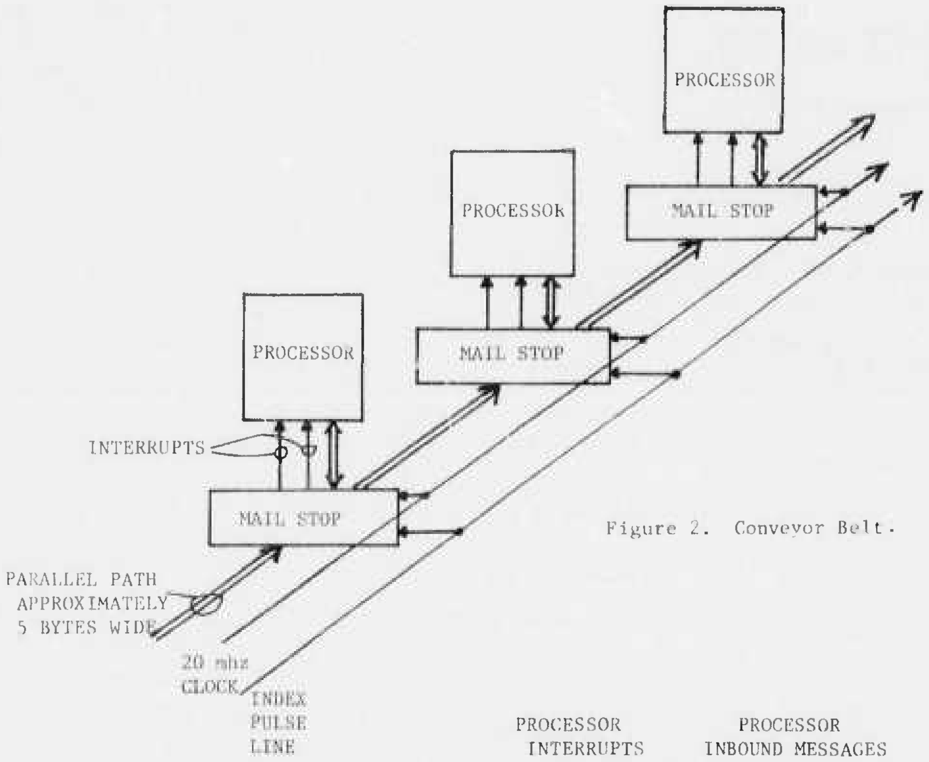


Figure 2. Convevor Belt.

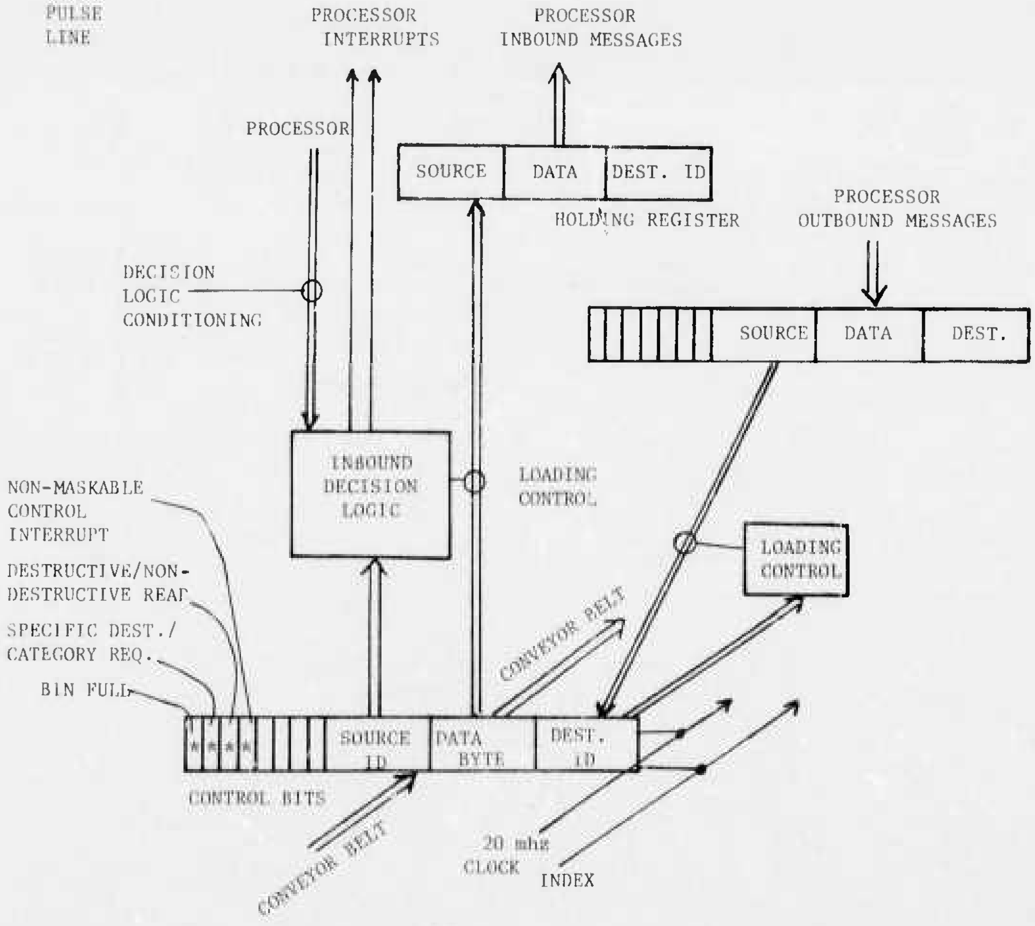


Figure 3. Mail Stop.

USING PYRAMIDS TO DEFINE LOCAL THRESHOLDS FOR BLOB DETECTION

Michael Shneier

Computer Vision Laboratory
Computer Science Center
University of Maryland
College Park, MD 20742

ABSTRACT

A method for detecting blobs in images is described. The method involves building a succession of lower resolution images and looking for spots in these images. A spot in a low-resolution image corresponds to a distinguished compact region in a known position in the original image. Further, it is possible to calculate thresholds in the low-resolution image, using very simple methods, and to apply those thresholds to the region of the original image corresponding to the spot. Examples are shown in which the technique is applied to several images.

1. INTRODUCTION

The most common way to extract objects from a picture is to threshold the picture. Many different techniques have been used to select good thresholds for this purpose [4]. Threshold selection involves choosing a gray level t such that all gray levels greater than t are mapped into the "object" label, while all other gray levels are mapped into the "background" label. In its simplest form, a single threshold is chosen for the whole image. This does not usually give good results because of variations in lighting, or because there are several objects in the picture with different gray-level characteristics. For better results, several local thresholds can be extracted from various parts of the picture, and can be applied just in those regions.

This paper describes a method of identifying parts of a picture on which to apply a threshold, and a means of calculating a local threshold for each of these parts. The method involves constructing a "pyramid" of images, each of lower resolution than its predecessor [1-3]. At some level of the pyramid, it is to be expected that any blob-like object should become spot-like. Thus, by running a spot-detector over the low-resolution images, the interesting regions in the picture can be discovered, and only these regions need be thresholded. In addition, the characteristics of the local regions (or the spots) can be used to calculate a good local threshold.

Examples are given of the application of the method to several images. In all cases the results are quite good, and highlight the usefulness of the method.

2. THE ALGORITHM

The algorithm has two main tasks. The first is to find parts of the picture that differ significantly from the background (likely objects), while the second is to calculate a local threshold for each of these parts and apply it in the neighborhood of the parts. Both tasks make use of the pyramid of low-resolution images.

1. If the whole pyramid has been constructed, stop. Otherwise, read in the previous pyramid level (the picture, if this is the first iteration).
2. Build a new level (see below).
3. Apply a spot detector to the new level.
4. Evaluate the spots resulting from step 3 and find "good" spots (see below). If there are too many good spots, go to 1.
5. For each good spot,
    a. calculate a threshold (see below);
    b. apply the threshold to the region in the original picture corresponding to the spot and write the results to the output picture.
6. Go to 1.

The original image forms the base of the pyramid. Each level is constructed on top of its predecessor, and is processed before its successor is constructed. This means that only one level need be maintained at any time, in addition to the original picture and the partially-constructed thresholded picture.

A pyramid level is constructed from its predecessor by mapping 2 by 2 squares of pixels from the previous level into single pixels in the new level. Two methods of calculating the new value from the old were implemented. The first involves simple averaging of the four pixels. In the second method, each 2 by 2 block of pixels is examined and the four gray levels are sorted in order of brightness. The middle two values are then averaged to give the new pixel corresponding to the 2 by 2 block. This process gives results that maintain edges reasonably well. In practice, both methods usually produce the same results. The new level of the pyramid is one quarter the size of the old (Figure 1a).

Having built a level of the pyramid, the next step is to apply a spot detector to it. The spot detector is a simple mask (Figure 2) that is applied at every point in the image. It looks for points that differ from their neighbors and scores them according to how much they differ. Note that the central value in the mask is smaller than an unbiased mask would require. This is to insure that the spots are more than marginally different from their neighbors. It tends to ignore spots caused by noise. The result of running the spot detector is a new image with high values where there are spots, and low values elsewhere.

The spot detector is very conservative, so another process is run to find a subset of "good" spots. Good spots are spots that are isolated. At low levels of the pyramid (high resolution), spots that are close together are deleted because they can be expected to merge into single spots higher up in the pyramid. At higher levels of the pyramid, this is not such a good idea because single spots represent large regions in the original picture. Thus, the definition of "good" is weighted by the level of the pyramid. A spot is good if the number of its neighbors that also responded positively to the spot detector is less than a level-dependent threshold.

Each spot in the low-resolution image corresponds to a region in the picture. If there are too many spots, then large parts of the picture will be covered. If there is indeed an object in the picture, it should coalesce into a smaller number of spots higher in the pyramid. If there is no object, then all the spots represent noise. In either case, the picture is too "busy". A maximum number of good spots is allowed at each level. If this number is exceeded, no further processing is performed, and a new pyramid level is constructed.

When a small enough number of good spots is discovered at a given level in the pyramid, the thresholding can be performed. Notice that it need only be applied to the regions in the picture corresponding to the spots in the pyramid. All other regions are ignored.

Many threshold selection techniques are applicable at this stage. There are the standard techniques [4] which may be applied to the picture itself in the region corresponding to a spot. In addition, it is possible to make use of the information in the low-resolution image to calculate a threshold. Both approaches were followed for the examples to be discussed here. Using the low-resolution image has the advantage that simple operations on the low resolution image correspond to complex operations involving much larger numbers of points in the picture.

The simplest threshold that can be extracted from the low resolution image is simply the gray level of the region in the picture corresponding to the spot. Usually, this threshold does not extract the whole object because the high gray levels bias the threshold, and there are very few non-object points in the region to provide an opposite bias (Figure 1c).

An alternative threshold is obtained by ignoring the spot itself and averaging the surrounding points in the low-resolution picture. This suffers from the opposite problem from the previous method. Now, too many non-object points reduce the threshold, and so parts of the background are classified as belonging to the object (Figure 1d).

A compromise between these two methods gives very good results. The outputs from the above two threshold selection processes are averaged, and the result is used as the threshold (Figure 1b).

The threshold is applied to a region slightly larger than that corresponding to the spot. This is to insure that parts of the object that were averaged into different points in the low-resolution image still may be classified, provided that they are not too far away from the spot center. If, indeed, the object extends a significant distance from the spot center, the spot detector should have found several spots in the neighborhood, each of which would be processed separately (or they would all be merged into a larger spot at the next level).

Another method of calculating a local threshold was also implemented. The method involves computing a histogram of the gray levels in the regions of the original picture that correspond to spots. For each spot a histogram is constructed for a region slightly larger than the projection of the spot onto the picture. The histogram is then examined, and a threshold is selected. The process of selection is complicated by the shape of the histogram, which tends either to be unimodal, or to have no significant peaks (Figure 3). The method that was used to find a threshold involves making an initial estimate, and refining the estimate on the basis of the shape of a part of the histogram.

The initial guess that was used was one of the naive thresholds mentioned above. The gray level corresponding to the spot in the pyramid provides an estimate of the gray level in the center of the object. Usually, the estimate needs to be modified to take account of parts of the object close to the background. To accomplish this, the histogram is examined, starting at the initial estimate, and moving in the direction of the background gray levels. The highest peak in the histogram in this direction is discovered, and the final threshold is chosen at the deepest valley between this peak and the initial estimate. This usually results in a good threshold, in most cases in one very similar to the averaging of the center and surround points in the pyramid discussed above.

The output picture is initially blank. The only regions of the picture that are changed are those that correspond to positive responses to the spot detector at some level in the pyramid. As a result, very little background noise appears in the output.

3. EXAMPLES

The method was applied to 24 FLIR images and to a picture of part of a handwritten signature. The results are shown in Figures 4-7. The examples are divided into three categories.

The first set of pictures (Figure 4) was processed using a simple averaging scheme for building the pyramids. The threshold was selected from the low resolution image by taking the average of the center (spot) gray level, and the average surrounding gray level.

Sometimes, when the contrast between the object and the background is small, the averaging process may cause the object to merge into the background. For FLIR imagery, it was found that it is often better to use the median instead of the average in building the pyramids. Figure 5 shows a set of examples where this was done. The threshold selection used the same method as for Figure 4.

The alternative method of selecting a threshold by examining the histogram is illustrated in Figures 6 and 7. Figure 6 shows four FLIR images and the results of thresholding them. The pyramids for these images were constructed by averaging, and the thresholds were selected by examining a histogram of a region in the image slightly larger than that corresponding to the spot.

Figure 7 illustrates the difference between selecting the threshold using only the low-resolution image, and making use of the histogram as well. For the signature in Figure 7, the histogram method results in a much cleaner thresholded image.

4. DISCUSSION

The blob-detection system described here is the first stage in a more ambitious feature-detection scheme. As it stands, the system provides a good threshold selection technique, with several advantages. One of the most important advantages is the ability of the system to isolate significant regions in a picture. This results both in better local threshold selection and in cleaner thresholded images. The thresholds are tailored specifically to the region to which they are applied, and uninteresting regions are ignored.

A problem that arose from the way the algorithm was implemented concerns the treatment of points on the borders of the picture. These points were ignored in the implementation, and, as a result, the algorithm discovered significant objects only if they were not on the border of the picture. This effect could be aggravated by the pyramid-building process because a point on the border of an image high in the pyramid corresponds to a fairly large region in the picture. There are several ways of overcoming this problem. For example, one-sided spot detectors could be used at the edges of the pictures, or the pictures could be extended either by reflection about the edge, or by folding the edges over so that the left and right and the top and bottom edges are contiguous.

A question that arises naturally concerns the amount of averaging between levels in the pyramid. Perhaps the exponential tapering used in these experiments is too harsh, and spot detectors of various intermediate sizes should be used in addition to those used here. This would more accurately capture the fine detail of the shapes and allow greater control over threshold selection. It is expected that further research will be conducted on this aspect of the algorithm.

An extension of the method that is currently under investigation is the detection of elongated objects. In conventional thresholding schemes, the shape of an object can only be discovered after the object has been extracted. It is not possible to search for objects with specific shape properties. Using the current method, however, it is possible to extract only those features that are of the desired shape. For example, to extract elongated objects, a line or streak detector can be applied instead of a spot detector. Preliminary results suggest that a straightforward extension of the blob-detection system can be produced which will detect only the elongated objects in a picture. This will help to alleviate a problem that sometimes arises when objects are not sufficiently blob-like. In such cases, some parts of the object may not be covered by the projection of a spot, and only part of the object may be thresholded.

Eventually, the system is envisaged as having multiple cooperating parts. Several feature detectors will be run at each level of the pyramid, for example, both line detectors and spot detectors. These would then interact within the levels and across levels. The whole system should be able to detect many different features simultaneously, and classify them on the basis of both local and global information.

5. CONCLUSIONS

A new method of detecting blobs in a picture by spot detection and local thresholding has been presented. The examples showed how simple threshold-detection calculations on low-resolution images can lead to good segmentation of the picture.

The method readily lends itself to extensions to more complex feature detection tasks, including detection of objects with specific properties, e.g. elongated objects.

It is expected that the method will eventually be included in a comprehensive, multilevel feature-extraction system that makes use of multiple-resolution images and responses from several different feature detectors.

References

1. A. R. Hanson and E. M. Riseman, Segmentation of natural scenes. In Computer Vision Systems (Hanson and Riseman, eds.), Academic Press, New York, pp. 129-163, 1978.

2. S. L. Tanimoto, Regular hierarchical image and processing structures in machine vision. Ibid., pp. 165-176.

3. L. Uhr, "Recognition cones," and some test results; the imminent arrival of well-structured parallel-serial computers; positions, and positions on positions. Ibid., pp. 363-378.

4. J. S. Weszka, A survey of threshold selection techniques. Computer Graphics Image Processing, 7, pp. 259-265, 1978.

± 1	± 1	± 1
± 1	∓ 7	± 1
± 1	± 1	± 1

Figure 2

The mask used for the spot detector.

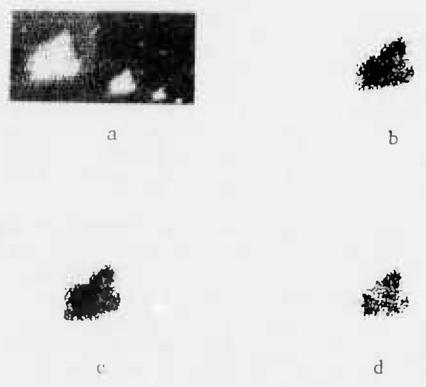


Figure 1

a) A FLIR image of a tank, and the pyramid constructed from it. b) Thresholded image using the average of center and surrounding spots.
c) Thresholded image using surrounding spots only.
d) Thresholded image using center spot only.

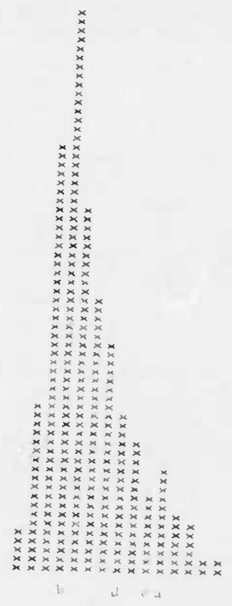


Figure 3

An example of a histogram used for threshold selection. Point a is the initial point chosen for thresholding (see text). Point b is the highest peak in the direction of the background. Point c is the point chosen as the final threshold. Point d is the threshold chosen by the method of averaging the center and background points in the low-resolution image. The histogram is for a spot in the bottom left picture of Figure 6. The small size of the spot results in a very low peak in the histogram (at a).

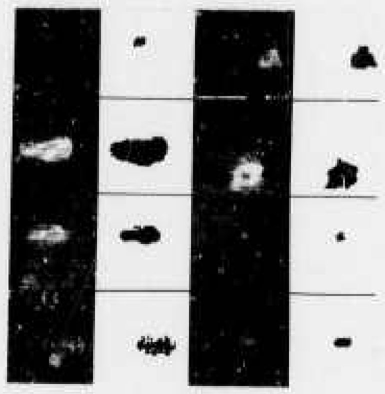


Figure 4

Eight FLIR images and their thresholded outputs. The pyramid was built by averaging in these examples and the threshold was selected as the average of the center and surrounding points in the low-resolution image.

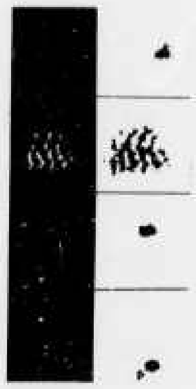


Figure 6

Four FLIR images and their thresholded outputs. The pyramids in these examples were constructed by averaging, and the threshold was selected by examining the histograms of local regions corresponding to spots.

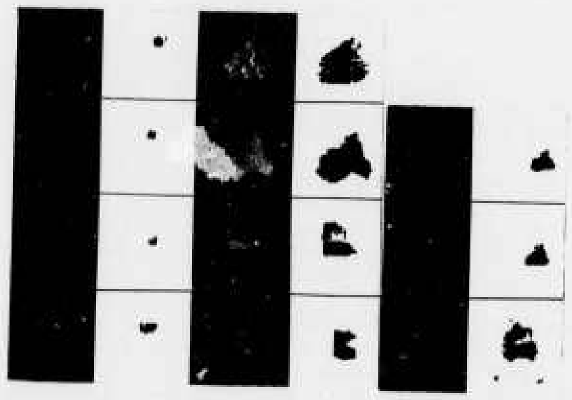


Figure 5

Eleven FLIR images and their thresholded outputs. The pyramid was built using the median and the threshold was selected as the average of the center and surrounding points in the low-resolution image.

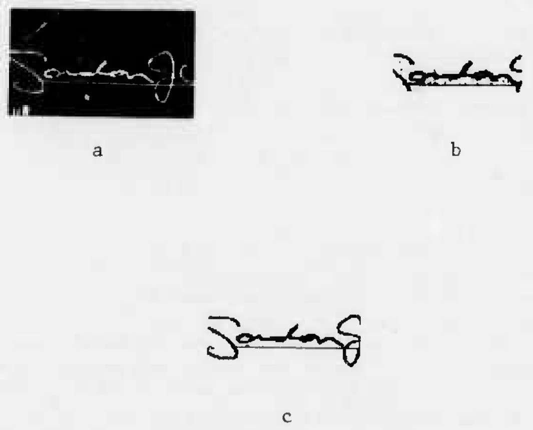


Figure 7

a) A picture of part of a handwritten signature.
b) The thresholded output using the average of the center and surrounding low-resolution points.
c) The result of calculating a threshold by examining the histogram.

QUADTREE STRUCTURES FOR REGION PROCESSING

Hanan Samet
Azriel Rosenfeld

Department of Computer Science
Computer Science Center
University of Maryland
College Park, MD 20742

ABSTRACT

Research into the use of the quadtree data structure for image processing applications is described. A quadtree represents an image array by a tree of out degree 4 which is constructed by recursively subdividing the array into blocks of constant value. This representation is particularly useful when applied to binary arrays representing regions (i.e., 1's are region points). Algorithms are informally discussed for conversion between this and other representations, and for measuring geometric properties of regions represented in this manner. Results of execution time analyses of these algorithms are also given.

1. INTRODUCTION

Region representation is an important issue in image processing, computer graphics and cartography. There are numerous representations currently in use. In this paper we focus our attention on the quadtree [1,6-11] representation. We discuss its relationship to more traditional representations and present informal descriptions of algorithms for converting between quadtrees and these representations. We also show how geometric properties of regions represented by quadtrees can be measured.

In our discussion we assume that a region is a subset of a 2^n by 2^n array which is viewed as being composed of unit-square pixels. The most common region representations used in image processing are the binary array and the run length representation [14]. The binary array represents region pixels by 1's and non-region pixels by 0's. The run length representation represents each row of the binary array as a sequence of runs of 1's alternating with runs of 0's.

Boundaries of regions are often specified as a sequence of unit vectors in the principal directions. This representation is termed a chain code [5]. For example, letting i represent 90°* i (i=0,1,2,3), we have the following sequence as the chain code for the region in Figure 1a:

$0\ 3\ 0^2\ 3^5\ 2^3\ 1\ 2\ 3^3\ 0\ 3\ 2^5\ 1^6\ 0\ 1\ 0\ 1\ 0\ 3\ 0\ 1\ 0\ 1$

Note that this is a clockwise code which starts at the leftmost of the uppermost border points. Chain codes yield a compact representation; however, they are somewhat inconvenient for performing operations such as set union and intersection.

Regions can also be represented by a collection of maximal blocks that are contained in the given region. One such trivial representation is the run length where the blocks are 1 by m rectangles. A more general representation treats the region as a union of maximal blocks (of 1's) of a given shape. The medial axis transform (MAT) [2,12] is the set of points serving as centers of these blocks and their corresponding radii.

The quadtree is a variant on the maximal block representation in which the blocks have standard sizes and positions (i.e., powers of two). It is an approach to region representation which is based on the successive subdivision of an image array into quadrants. If the array does not consist entirely of 1's or entirely of 0's, then we subdivide it into quadrants, subquadrants,... until we obtain blocks (possibly single pixels) that consist of 1's or of 0's, i.e., they are entirely contained in the region or entirely disjoint from it. This process is represented by a tree of out degree 4 in which the root node represents the entire array. The four sons of the root node represent the quadrants (labeled in order NW, NE, SW, SE), and the leaf nodes correspond to those blocks of the array for which no further subdivision is necessary. Leaf nodes are said to be "black" or "white" depending on whether their corresponding blocks are entirely within or outside of the region respectively. All non-leaf nodes are said to be "gray". Since the array was assumed to be 2^n by 2^n, the tree height is at most n. As an example, Figure 1b is a block decomposition of the region in Figure 1a while Figure 1c is the corresponding quadtree.

2. PRELIMINARIES

In the quadtree representation, by virtue of its tree-like nature, most operations are carried out by techniques which traverse the tree. In fact, many of the operations that we describe can be characterized as having two basic steps. The first step either traverses the quadtree in a specified order or constructs a quadtree. The second step performs a computation at each node which often makes use of its neighboring nodes, i.e. nodes representing image blocks that are adjacent to the given node's block. Frequently these two steps are performed in parallel.

In general, we prefer to avoid having to use position (i.e., coordinates) and size information when making relative transitions (i.e., locating neighboring nodes) in the quadtree since they involve computation (rather than simply chasing links) and are clumsy when adjacent blocks are of different sizes (e.g., when a neighboring block is larger). Also, we do not assume that there are links from a node to its neighbors, because we do not want to use links in excess of four links from a non-leaf node to its sons and the link from a non-root node to its father. Thus all of our operations are implemented by algorithms that make use of the existing structure of the tree. This is in contrast with the methods of Klinger and Rhodes [11] which make use of size and position information, and those of Hunter and Steiglitz [6-8] which locate neighbors through the use of explicit links (termed nets and ropes).

Locating neighbors in a given direction is quite straightforward. Given a node corresponding to a specific block in the image, its neighbor in a particular direction (horizontal or vertical) is determined by locating a common ancestor. For example, if we want to find a eastern neighbor, the common ancestor is the first ancestor node which is reached via its NW or SW son. Next, we retrace the path from the common ancestor, but making mirror image moves about the appropriate axis, e.g., to find an eastern or western neighbor, the mirror images of NE and SE are NW and SW, respectively. For example, the eastern neighbor of node 32 in Figure 1c is node 33. It is located by ascending the tree until the common ancestor, H, is found. This requires going through a SE link to reach L and a NW link to reach H. Node 33 is now reached by backtracking along the previous path with the appropriate mirror image moves (i.e., going through a NE link to reach M and a SW link to reach 33).

In general, adjacent neighbors need not be of the same size. If they are larger, then only a part of the path to the common ancestor is retraced. If they are smaller, then the retraced path ends at a "gray" node of equal size. Note that similar techniques can be used to locate diagonal neighbors (i.e., nodes corresponding to blocks that touch the given node's block at a corner). For example, node 20 in Figure 1c is the NW neighbor of node 22. For more details, see [21].

3. CONVERSION

3.1 Quadtrees and Arrays

The definition of a quadtree leads naturally to a "top down" quadtree construction process. This may lead to excessive computation because the process of examining whether a quadrant contains all 1's or all 0's may cause certain parts of the region to be examined repeatedly by virtue of being composed of a mixture of 1's and 0's. Alternatively, a "bottom-up" method may be employed which scans the picture in the sequence

```
 1  2  5  6 17 18 21 22
 3  4  7  8 19 20 23 24
 9 10 13 14 25 26 29 30
11 12 15 16 27 28 31 32
33 ...
```

where the numbers indicate the sequence in which the pixels are examined. As maximal blocks of 0's or 1's are discovered, corresponding leaf nodes are added along with the necessary ancestor nodes. This is done in such a way that leaf nodes are never created until they are known to be maximal. Thus there is never a need to merge four leaves of the same color and change the color of their common parent from gray to white or black as is appropriate. See [19] for the details of such a algorithm whose execution time is proportional to the number of pixels in the image.

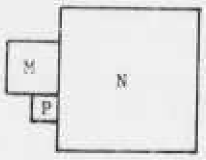
If it is necessary to scan the picture row by row (e.g., when the input is a run length coding) the quadtree construction process is somewhat more complex. We scan the picture a row at a time. For odd-numbered rows, nodes corresponding to the pixel or run values are added to the tree, one node per pixel. For even-numbered rows, nodes are added for the pixels and attempts are made to discover maximal blocks of 0's or 1's whose size depends on the row number (e.g., when processing the fourth row, maximal blocks of maximum size 4-by-4 can be discovered). In such a case merging is said to take place. See [18] for the details of an algorithm that constructs a quadtree from a row by row scan such that at any instant of time a valid quadtree exists. This algorithm has an execution time that is proportional to the number of pixels in the image.

Similarly, for a given quadtree we can output the corresponding binary picture by traversing the tree in such a way that for each row the appropriate blocks are visited and a row of 0's or 1's is output. In essence, we visit each quadtree node once for each row that intersects it (i.e., a node corresponding to a block of size 2^K by 2^K is visited 2^K times). For the details see [20] where an algorithm is described whose execution time depends only on the number of blocks of each size that comprise the image - not on their particular configuration.

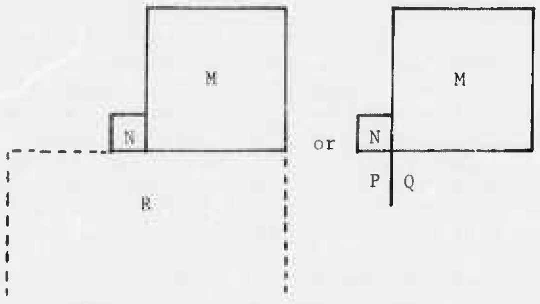
3.2 Quadtrees and Borders

In order to determine, for a given leaf node M of a quadtree, whether the corresponding block is on the border, we must visit the leaf nodes that correspond to 4-adjacent blocks and check whether they are black or white. For example, to find M's right hand neighbor we use the neighbor finding techniques outlined in Section 2. If the neighbor is a leaf node, then its block is at least as large as that of M and so it is M's sole neighbor to the right. Otherwise, the neighbor is the root of a subtree whose leftmost leaf nodes correspond to M's right-hand neighbors. These nodes are found by traversing that subtree.

Let M,N be black and white leaf nodes whose associated blocks are 4-adjacent. Thus the pair M,N defines a common border segment of length 2^K (2^K is the minimum of the side lengths of M and N) which ends at a corner of the smaller of the two blocks (they may both end at a common point). In order to produce a boundary code representation for a region in the image we must determine the next segment along the border whose previous segment lay between M and N. This is achieved by locating the other leaf P whose block touches the end of the segment between M and N:



If the M,N segment ends at a corner of both M and N, then we must find the other leaf R or leaves P,Q whose blocks touch that corner:



Again, this can be accomplished by using neighbor finding techniques as outlined in Section 2.

For the non-common corner case, the next border segment is the common border defined by M and P if P is white, or the common border defined by N and P if P is black. In the common corner case, the pair of blocks defining the next border segment is determined exactly as in the standard "crack following" algorithm [13] for traversing region borders. This process is repeated until we re-encounter the block pair M,N. At this point the entire border has been traversed. The successive border segments constitute a 4-direction chain code, broken up into segments whose lengths are sums of powers of two. The time required for this process is on the order of the number of border nodes times the tree height. For more details see [4].

Using the methods described in the last two paragraphs, we can traverse the quadtree, find all borders, and generate their codes. During this process, we mark each border as we follow it, so that it will not be followed again from a different starting point. Note that the marking process is complicated by the fact that a node's block may be on many different borders.

In order to generate a quadtree from a set of 4-direction chain codes we use a two-step process. First, we trace the boundary in a clockwise direction and construct a quadtree whose black leaf nodes are of a size equal to the unit code length. All the black nodes correspond to blocks on the interior side of the boundary. All remaining nodes are left uncolored. Second, all uncolored nodes are set to black or white as appropriate. This is achieved by traversing the tree, and for each uncolored leaf node, examining its neighbors. The node is colored black unless any of its neighbors is white or is black with a border along the shared boundary. At any stage, merging occurs if the four rows of a non-leaf node are leaves having the same color. The details of the algorithm are given in [15]. The time required is proportional to the product of the perimeter (i.e., the 4-direction chain code length) and the tree height.

3.3 Quadtrees of Derived Sets

Let S be the set of 1's in a given binary array, and let $\overline{S}$ be the complement of S. The quadtree of $\overline{S}$ is the same as that of S, with black leaf nodes changed to white and vice versa. To get the quadtree of $S \cup T$ from those of S and T, we traverse the two trees simultaneously. Where they agree, the new tree is the same and if the two nodes are gray, then their subtrees are traversed. If S has a gray (=nonleaf) node where T has a black node, the new tree gets a black node; if T has a white node there, we copy the subtree of S at that gray node into the new tree. If S has a white node, we copy the subtree of T at the corresponding node. The algorithm for $S \cap T$ is exactly analogous, with the roles of black and white reversed. The time required for these algorithms is proportional to the number of nodes in the smaller of the two trees [23].

3.4 Skeletons and Medial Axis Transforms

The medial axis of a region is a subset of its points each of which has a distance from the complement of the region (using a suitably defined distance metric) which is a local maximum. The medial axis transform (MAT) consists of the set of medial axis or "skeleton" points and their associated distance values. The quadtree representation may be rendered even more compact by the use of a skeleton-like representation. Recall that a quadtree is a set of disjoint maximal square blocks having sides whose lengths are powers of 2. We define a quadtree skeleton to be a set of maximal square blocks having sides whose lengths are sums of powers of two. The maximum value (i.e., "chessboard") distance metric [13] is the most appropriate for an image represented by a quadtree. See [21] for the details of its computation for a quadtree; see also [24] for a different quadtree distance transform. A quadtree medial axis transform (QMAT) is a quadtree whose black nodes correspond to members of the quadtree skeleton while all remaining leaf nodes are white. See [22] for the details of a quadtree to QMAT conversion algorithm whose execution time is on the order of the number of nodes in the tree.

4. PROPERTY MEASUREMENT

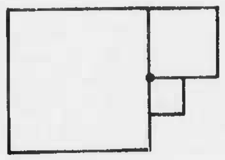
4.1 Connected Component Labeling

Traditionally, connected component labeling is achieved by scanning a binary array row by row from left to right and labeling adjacencies that are discovered to the right and downward. During this process equivalences will be generated. A subsequent pass merges these equivalences and updates the labels of the affected pixels. In the case of the quadtree representation we also scan the image in a sequential manner. However, the sequence's order is dictated by the tree structure - i.e., we traverse the tree in postorder. Whenever, a black leaf node is encountered all black nodes that are adjacent to its south and east sides are also visited and are labeled accordingly. Again, equivalences generated during this traversal are subsequently merged and a tree traversal is used to update the labels. The interesting result is that the algorithm's execution time is proportional to the number of blocks in the image and does not depend on their size. In contrast, for the binary array representation the execution time is proportional to the number of pixels. An analogous result is described in the next section. See [17] for the details of an algorithm that labels connected components in time on the order of the number of nodes in the tree plus the product of $B \cdot \log B$ where B is the number of black leaf nodes.

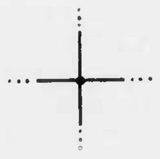
4.2 Component Counting and Genus Computation

Once the connected components have been labeled, it is trivial to count them, since their number is the same as the number of inequivalent labels. We will next describe a method of determining the number of components minus the number of holes by counting certain types of local patterns in the array; this number, g, is known as the genus or Euler number of the array.

Let V be the number of 1's, E the number of 11's and $\begin{smallmatrix}1\\1\end{smallmatrix}$'s, and F the number of $\begin{smallmatrix}11\\11\end{smallmatrix}$'s in the array; it is well known [13] that $g=V-E+F$. This result can be generalized to the case where the array is represented by a quadtree [3]. In fact, let V be the number of black leaf nodes; E the number of pairs of such nodes whose blocks are horizontally or vertically adjacent; and F the number of triples or quadruples of such nodes whose blocks meet at and surround a common point, e.g.



or



Then $g=V-E+F$. These adjacencies can be found (see Section 3.2) by traversing the tree; the time required is on the order of the number of nodes in the tree.

4.3 Area and Moments

The area of a region represented by a quadtree can be obtained by summing the areas of the black leaf nodes, i.e., counting 4^h for each such node that represents a 2^h by 2^h block. Similarly, the first x and y moments of the region relative to a given origin can be computed by summing the first moments of these blocks; note that we know the position (and size) of each block from the coordinates of its leaf in the tree. Knowing the area and the first moments gives us the coordinates of the centroid, and we can then compute central moments relative to the centroid as the origin. The time required for any of these computations is proportional to the number of nodes in the tree. Further details on moment computation from quadtrees can be found in [23].

4.4 Perimeter

An obvious way of obtaining the perimeter of a region represented by a quadtree is to simply traverse its border and sum the number of steps. However, there is no need to traverse the border segments in order. Instead, we use a method which traverses the tree in postorder and for each black leaf node examines the colors of its neighbors on its four sides. For each white neighbor the length of the corresponding border segment is included in the perimeter. See [16] for the details of such an algorithm which has execution time proportional to the number of nodes in the tree.

5. CONCLUDING REMARKS

We have briefly sketched algorithms for accomplishing traditional region processing operations by use of the quadtree representation. Many of the methods used on the pixel level carry over to the quadtree domain (e.g., connected component labeling, genus, etc.). Because of its compactness, the quadtree permits faster execution of these operations. Often the quadtree algorithms require time proportional to the number of blocks in the image, independent of their size.

Quadtrees constitute an interesting alternative to the standard methods of digitally representing regions. Their chief disadvantage is that they are non shift-invariant; two regions differing only by a translation may have quite different quadtrees (but see [22]). Thus shape matching from quadtrees is not straightforward. Nevertheless, in other respects they have many potential advantages. They provide a compact and easily constructed representation from which standard region properties can be efficiently computed. In effect, they are "variable-resolution arrays" in which detail is represented only when it is available, without requiring excessive storage for parts of the image where detail is missing.

References

1. N. Alexandridis and A. Klinger, Picture decomposition, tree data-structures, and identifying directional symmetries as node combinations, Computer Graphics Image Processing 8, 1978, 43-77.

2. H. Blum, A transformation for extracting new descriptors of shape, in W. Wathen-Dunn, ed., Models for the Perception of Speech and Visual Form, M.I.T. Press, Cambridge, MA, 1967, 362-380.

3. C. R. Dyer, Computing the Euler number of an image from its quadtree, Computer Science TR-769, University of Maryland, College Park, MD, May 1979, to appear in CGIP.

4. C. R. Dyer, A. Rosenfeld, and H. Samet, Region representation: boundary codes from quadtrees, Computer Science TR-732, University of Maryland, College Park, MD, February 1979, to appear in CACM.

5. H. Freeman, Computer processing of line-drawing images, Computing Surveys 6, 1974, 57-97.

6. G. M. Hunter, Efficient computation and data structures for graphics, Ph.D. dissertation, Dept. of Electrical Engineering and Computer Science, Princeton University, Princeton, NJ, 1978.

7. G. M. Hunter and K. Steiglitz, Operations on images using quad trees, IEEETPAMI-1, 1979, 145-153.

8. G. M. Hunter and K. Steiglitz, Linear transformations of pictures represented by quad trees, Computer Graphics Image Processing 10, 1979, 289-296.

9. A. Klinger, Data structures and pattern recognition, Proc. 1st Intl. Joint Conference on Pattern Recognition, 1973, 497-498.

10. A. Klinger and C. R. Dyer, Experiments in picture representation using regular decomposition, Computer Graphics Image Processing 5, 1976, 68-105.

11. A. Klinger and M. L. Rhodes, Organization and access of image data by areas. IEETPAMI-1, 1979, 50-60.

12. J. L. Pfaltz and A. Rosenfeld, Computer representation of planar regions by their skeletons, Comm. ACM 10, 1967, 119-122, 125.

13. A. Rosenfeld and A. C. Kak, Digital Picture Processing, Academic Press, New York, 1976.

14. D. Rutovitz, Data structures for operations on digital images, in G. C. Cheng et al., eds., Pictorial Pattern Recognition, Thompson Book Co., Washington, DC, 1968, 105-133.

15. H. Samet, Region representation: quadtrees from boundary codes, Computer Science TR-741, University of Maryland, College Park, MD, March 1979, to appear in CACM.

16. H. Samet, Computing perimeters of images represented by quadtrees, Computer Science TR-755, University of Maryland, College Park, MD, April 1979.

17. H. Samet. Connected component labeling using quadtrees, Computer Science TR-756, University of Maryland, College Park, MD, April 1979.

18. H. Samet, Region representation: raster-to-quadtree conversion, Computer Science TR-766, University of Maryland, College Park, MD, May 1979.

19. H. Samet, Region representation: quadtrees from binary arrays, Computer Science TR-767, University of Maryland, College Park, MD, May 1979, to appear in CGIP.

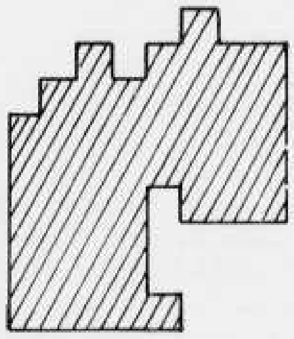
20. H. Samet, Region representation: quadtree-to-raster conversion, Computer Science TR-768, University of Maryland, College Park, MD, June 1979.

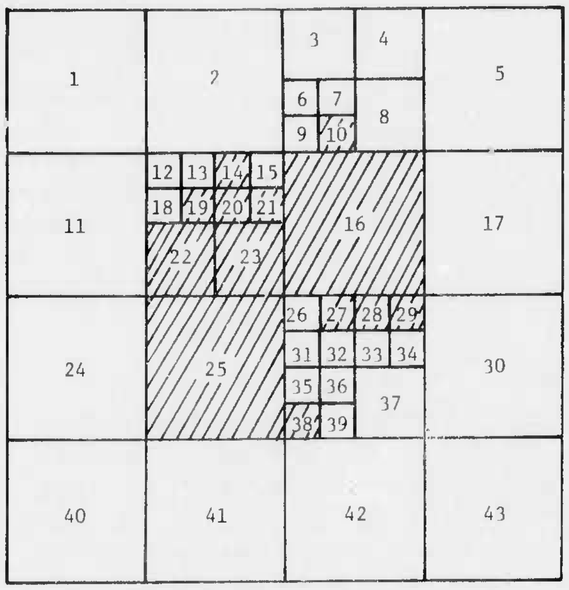
21. H. Samet, A distance transform for images represented by quadtrees, Computer Science TR-780, University of Maryland, College Park, MD, July 1979.

22. H. Samet, A quadtree medial axis transform, Computer Science TR-803, University of Maryland, College Park, MD, August 1979.

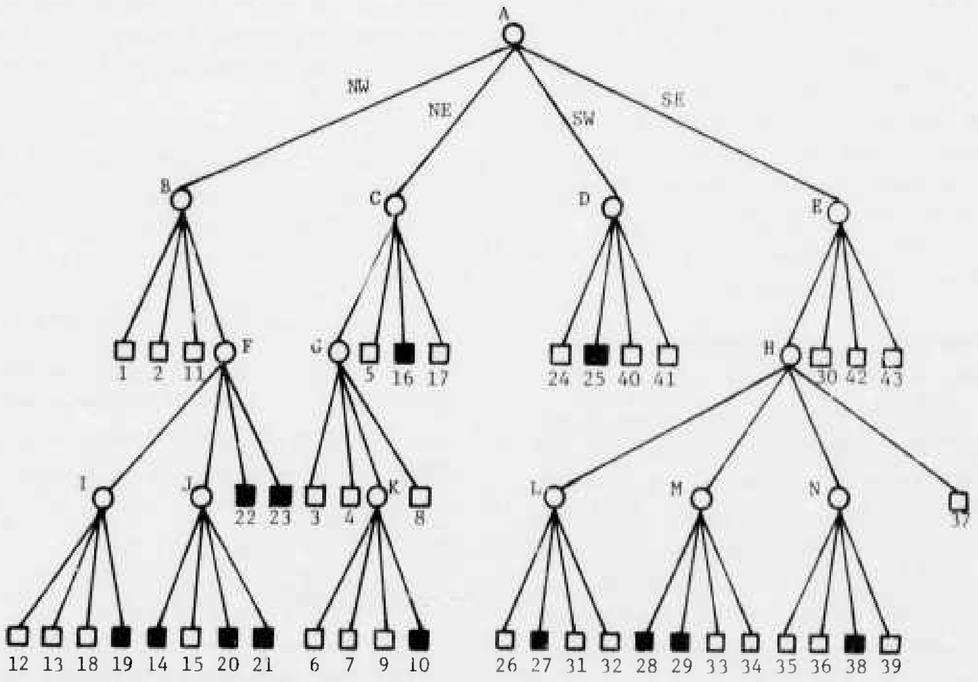
23. M. Shneier, Linear time calculation of geometric properties using quadtrees, Computer Science TR-770, University of Maryland, College Park, MD, May 1979.

24. M. Shneier, A path-length distance transform for quadtrees, Computer Science TR-794, University of Maryland, College Park, MD, July, 1979.


a. Region



b. Block decomposition of the region in (a).



c. Quadtree representation of the blocks in (b).

Figure 1. A region, its maximal blocks, and the corresponding quadtree. Blocks in the region are shaded, background blocks are blank.

SEARCH STRATEGIES FOR THE ARGOS IMAGE UNDERSTANDING SYSTEM

David R. Smith

Department of Computer Science
Carnegie-Mellon University
Pittsburgh, PA 15213

Summary

Argos is an image understanding system which was built to test the application of the Locus search technique to the task of image understanding. By a sequence of small modifications, Locus search can be transformed into a relaxation method. Relaxation using the Locus probability updating rules exhibits superior performance. We present some thoughts on how shape information can be applied.

1. Introduction

The Argos image understanding system[1, 2] attempts to explain an image by matching segments of the image to a set of labels. The segments may be individual pixels or contiguous regions designated by hand or by a segmenter program. The label set defines the set of real-world objects which may be identified by the system, e.g., Three Rivers Stadium, or Fort Pitt Bridge.

Image labelling, or the assignment of a label to each segment, is performed under the control of a knowledge network, which is a directed graph. Each node of the network describes a label, and contains one or more templates. A template describes the spectral and shape properties expected of an area which is to be given the label. Usually a node contains only one template. Multiple templates cope with objects which have radically varying appearance, such as the Allegheny River with and without ice.

The arcs of the network represent possible adjacencies of segments with differing labels. We use just the coarse directions "left-of" and "above", plus their inverses. Presence of an arc from label A to label B in direction δ indicates that it is possible for the image to contain object A in the direction δ from object B. The presence of an arc indicates merely that the transition permission is nonzero. We usually think of transition permissions as 0-1 valued, but fractional values can be used to indicate a self-transition penalty or bonus.

Using the local knowledge contained in the network nodes, a score vector is computed for each segment, telling how well it matches the most favorable template of each label. From this match vector is computed a normalized probability vector, for which each entry indicates the probability that the particular segment has a given label.

The job of the search algorithm is to find the overall pairings of segments to labels which best fits the *a priori* probability vectors, consistent with the connectivity arcs of the network.

2. Search Strategies

The search strategies which we have employed with Argos lie along a spectrum which includes Locus search at one end and relaxation at the other. We now quickly review the principal ideas of the two methods.

2.1 Relaxation

The relaxation method repeatedly updates in parallel the per-segment label probability vectors. The updating function for a segment takes into consideration the segment's present label probability vector, the vectors of the segment's neighbors, and the transition permissions along the directions separating the segments. In short,

$$\overline{P_{sj}}^{(i+1)} = P_{sj}^{(i)} * \underset{n \in N}{\text{Avg}} \underset{k \in L}{\text{Avg}} P_{nk}^{(i)} * T_{kj\delta} \qquad (1)$$

where

s	is the segment for which the vector is being recomputed.
j	is a candidate label for segment s.
$P_{(sj)}^{(i)}$	is the probability as computed in iteration i that segment s has label j. (The barred value is what is developed prior to renormalization.)
N	is the set of segments which are neighbors to s.
n	is an individual neighbor.
L	is the set of labels.
k	is a label of a neighbor segment.
δ	is the direction from which neighbor n is separated from segment s.
T_δ	is the transition permission from label k to label j in direction δ.

The relaxation process is iterated until a concensus develops on a label for each segment.

2.2 Locus

Locus search has its roots in dynamic programming. For a one-dimensional signal, it amounts to dynamic programming with pruning of poor candidates. Locus was used with success in the Harpy speech understanding system. [3] Dynamic programming is strictly valid only for one-dimensional signals such as speech waveforms. But one of its premises suggested that it might be profitably employed on a two-dimensional image signal. The premise is that in the globally best labelling, the labels chosen must be consistent with each other, but not necessarily with runners up. This statement may not seem startling, but it allows a reduction in the computational complexity of the search. The updating function is the same as noted above, save that the second average is changed to a maximum:

$$\overline{P_{sj}}^{(i+1)} = P_{sj}^{(i)} * \underset{n \in N}{\text{Avg}} \underset{k \in L}{\text{Max}} P_{nk} * T_{kj\delta} \qquad (2)$$

The straightforward transmutation of Locus into a 2-D algorithm retains the "one-and-a-fraction pass" nature of its predecessor, as well as back pointers. The algorithm proceeds as follows.

An ordering is induced on the segments, such that insofar as possible, each segment has neighbors which precede it, and neighbors which follow it. These are called predecessor and successor neighbors, respectively.

One pass is made through the ordered segment list to apply contextual information to update each segment's label probabilities. The context used for a segment is its predecessors, which already have undergone reprocessing. In this way, the network constraints propagate forward through the segment list. Thus, the updating of the last segment takes into account all that has gone before. During this forward pass, the computation of eq. 2 implicitly selects a best choice for the label of each predecessor. The best choice for a neighbor n is the label k which establishes the value of the Max operator. This value of k is recorded as the *backpointer* to neighbor n from label j of segment s. Informally, the backpointer states, "if segment s is labelled j, then the best label for n is k."

These backpointers are used to propagate contextual information in the reverse direction. After the forward pass, when all backpointers have been computed, the best global labelling can be reconstructed by following the chain of backpointers, beginning with the best label of the last segment.

The fly in the ointment of pure Locus is that most segments have multiple successors, and that the successors' backpointers tend to disagree. Worse than sheer disagreement is the case when the label proposed by one successor is incompatible with the final labelling of another successor.

Originally, Argos coped with such problems by discounting the backpointers which would cause conflicts, and then by plurality vote if necessary. This proved to be unsatisfactory, because it throws away hard-won summary information. This is quite serious, because conflicts turn out to be very frequent.

It is clear that in such circumstances, the successors must negotiate the best mutually acceptable compromise. This can be easily achieved by applying forward pass techniques to the back pass. The pointer traceback is replaced by a *backsearch* which computes a segment's final label. It starts with the scores left over from the forward pass, and applies network consistency constraints to the final labellings of the successor segments.

This "backsearch" version of Locus results in a more robust system. Its labelling accuracy is better, and it supports its findings more solidly.

Argos has a mechanism to prune segment-label candidates which score poorly. The mechanism is activated when a probability is below some threshold which is placed relative to the best present candidate. Backsearch is equivalent to running Locus backward through the data with the relative pruning threshold set to zero.

The next search variant soon suggests itself: run the Locus updater back and forth through the segment list, slowly decreasing the relative pruning threshold to zero. This method is superior to the others.

At this stage, the algorithm actually is recognizable as a sequentially implemented relaxation process. To convert it to the classical variant, we can eliminate the distinction between predecessor and successor neighbors and update all of the probability vectors in parallel. Then the only distinction is whether the maximum operator or the average appears in the updating function. (It's a relaxation process either way.)

3. Speedups

It is worth considering how these algorithms differ in computational complexity, and what properties might be exploited to speed the search. We will consider pruning, sorting, and parallelism.

3.1 Pruning

Obviously, poor labels can be pruned with any of these search algorithms. It seems, however, that parallel relaxation must wait longer (more iterations) before pruning heavily. The sequential variants use contextual information from longer range on the first pass.

An interesting effect of pruning is that segments which in the early going emerge to have quite definite labels will have the other labels pruned away. Then they can't change through subsequent iterations. They cause the search to appear to be island-driven.

3.2 Sorting

The variants which use eq. 2 for the updating function can derive more benefit from keeping the probability vectors sorted. Most labels, even poorly scoring ones, will link to fairly high-scoring neighbor labels, so it pays to keep the high-scoring labels at the front of the list. If label probability vectors are kept sorted, then computing the Max transition consists of finding the first label in the list with a legal transition. In the case of multivalued transition permissions, a neighbor's probability vector must be examined only to the point of proving that no following value can improve the score. In contrast, updating function 1 requires that the neighbor's whole probability list must be taken into the updating calculation.

3.3 Parallelism

Locus has been criticized as being an inherently serial algorithm which could not take advantage of parallelism. It is true that segments are considered sequentially, but that only ties down variable s in equations 1 and 2. It may well be that s is the simplest place to parcel out different values to several processors, but there are other opportunities for parallelism. A different candidate label could be assigned to each processor, thus finding parallelism in the j variable. Similarly, the neighbor n and its label k could yield parallelism.

4. Shape

The shape knowledge which Argos maintains for each label is its orientation, elongation, and compactness. Compactness is the ratio of area to the square of the perimeter. Orientation and elongation are determined by computing the best-fit ellipse. From the map database, these shape measures are precomputed for each template at the time the positional constraint network is derived. For example, the U.S. Steel building will be found to be quite elongated in the vertical direction, so a segment or cluster of segments which exhibits vertical elongation would find support for being labelled "U.S. Steel building".

The shape information carries with it a modicum of occlusion knowledge. If the network is compiled for a viewpoint in which Mellon Bank partially occludes the U.S Steel building, then this will be reflected as a smaller size and different shape for the latter. The compiled network will know to expect the shape change in the image.

A problem arises with the orientation feature. For shapes with very low eccentricity, a small perturbation could cause a drastic change in the orientation. One solution is to place a low weight on the orientation measure when the elongation is low. This is achieved by considering eccentricity and orientation to be polar coordinates of a shape vector. Shape comparisons are achieved by taking the magnitude of the difference of the two shape vectors. For convenience, this shape vector can be converted from polar form to cartesian. Then the two components can be treated as separate features without either suffering from the pole problem that orientation does. The advantage of treating them separately is that it is easier to merge them into the general feature vector which describes the PPE.

The use of shape knowledge suffers from chicken-egg behavior, regardless of the particular labelling search strategy. The shape information is supposed to provide guidance for the local labelling of a segment. Yet, the shape match cannot be applied until it is known which set of contiguous segments is to be considered as a commonly-labelled group. If the shape is computed for only the current segment, or for it and its similar immediate neighbors, then there is still a good chance that some other segment should have been included, or that one of the included neighbors would have better been omitted. The first approach taken by Argos was to record, for each candidate label of a segment, a bit-map defining the contiguous set of segments which could be expected (based on backpointers) to receive the same label. The shape computation would be deferred until the process had proceeded to the last successor neighbor, so as to allow the bit maps to accumulate. This last neighbor would apply the shape match. Thus, the current segment's shape match would affect the last successor's view of its environment, and hence its label and backpointers. And the last successor's backpointers would affect the choice of labels for the current segment and others.

With relaxation labelling, we can wait for a couple of iterations before applying shape knowledge. This allows many segment-label pairs to be pruned, which reduces the number of combinations of segments for which shape measures must be found.

Our present test data has not allowed the quality of the shape procedures to be fully tested. With pixelwise labelling, the search space is too large: too many labels for too many segments have different ideas about where the boundaries lie for which shape can be computed. Hand-segmented images are generally not oversegmented, so for them shape evaluation usually involves only one segment at a time. This doesn't result in a fully satisfactory test. It does, however, turn up an interesting phenomenon. Although the shape measures provided good matches, the Locus search did not apply them frequently enough (only at the last successor). Sometimes shape would cause the last successor's backpointer to be correct, but the beneficial effect did not usually propagate back through mutual neighbors. So the result would be only to cause a backpointer conflict.

The present scheme for applying shape knowledge is not fully satisfactory, either theoretically or from the point of view of computational complexity. Shape is an area where much remains to be done.

5. Performance

The cross of Locus with relaxation seems to propagate influence from a distance more effectively than either pure Locus or the backsearch version. To illustrate its effectiveness: the present battery of experiments was run using two pictures of Pittsburgh as guinea pigs. One had 39 segments, of which 9 would be correctly identified using optical match alone. Pure Locus (with backpointers) obtained 14 correctly labelled segments, backsearch 21, and repeated passes 33. By biasing the network to discourage self-transitions, backsearch achieved 36, but this might constitute an adaptation to the particular image. It seems hardly likely that any search strategy would dig out the other three segments, since they have very bad signal matches.

The other image had 16 segments. All three methods correctly labelled 14 of them. However, backsearch and the multipass algorithm obtained their results quite solidly, whereas pure Locus had some lucky bounces.

The relaxation labeller requires about two to three times the processing time of pure Locus. This seems quite acceptable in view of the increase in accuracy.

References

1. Steven Rubin, *The ARGOS Image Understanding System*, PhD dissertation, Computer Science Department, Carnegie-Mellon University, 1976.

2. Steven Rubin, "The ARGOS Image Understanding System," *Proceedings of Image Understanding Workshop*, , November 1978, pp. 159.

3. Bruce T. Lowerre, *The HARPY Speech Recognition System*, PhD dissertation, Computer Science Department, Carnegie-Mellon University, 1976.

TEXTURE ENERGY MEASURES

Kenneth I. Laws

Image Processing Institute
University of Southern California
Los Angeles, California 90007

INTRODUCTION

This paper presents a set of "texture energy" transforms that provide texture measures for each pixel of a monochrome image. The transforms are fast, requiring only one-dimensional convolutions and simple moving-average techniques. The method is more accurate than gray level co-occurrence methods. It is local, operating on small image windows in much the same manner as the human visual system. It can be made invariant to changes in luminance, contrast, and rotation without histogram equalization or other preprocessing.

These texture measures are more local than previously studied frequency-domain statistics. Frequency components are measured with very small convolution masks, and phase relationships within each window are measured without regard to any global origin. This method, similar to human visual processing, is appropriate for textures with a short coherence length or correlation distance.

Figures 1a and 1b show the sequence of images, or image blocks, used in measuring texture. The original image is first filtered with a set of small convolution masks, typically 5x5 masks with integer coefficients. Only one-dimensional convolution is required, since the masks are separable. The filtering could also be accomplished optically or with multistage 3x3 convolutions.

The filtered images are then processed with a nonlinear "local texture energy" filter. This is simply a moving-window average of the absolute image values. Such moving-window operations are very fast even on general-purpose digital computers. The best window size depends on the size of image texture regions. This study has concentrated on 15x15 windows. Even smaller windows might be useful if color information were available.

The next step in Figure 1 shows the linear combination of texture energy planes into a smaller number of principal component planes, typically three or four. This is an optional data compression step. It is tempting to call the final images "perceptual planes," but it has not yet been proven that they relate to human texture perception. They do seem to represent natural texture dimensions, and to be more "reliable" than the texture energy planes.

The final output is a segmented or labeled image. A classifier assigning texture labels to the image pixels can take either texture energy planes or principal component planes as input. Classification is simple and fast if texture classes are known a priori. Clustering or segmentation algorithms must be used if texture classes are unknown.

Texture Data

In an experimental study, the results can be no better than the input data. We require a set of uniform texture fields large enough to provide adequate samples of each texture. Ideally this training set should come from a target application area. For a general vision system, each texture must be a "natural" one, and the set must include a range of natural texture dimensions. We avoid artificially generated textures, such as sinusoidal gratings, because they would favor the Fourier transform and other frequency domain measures.

The textures we have chosen are from high-resolution photographs used in the Brodatz texture album. Figure 2a shows a composite. The first two rows of 128x128 blocks are from the images of Grass, Raffia, Sand, Wool, Pigskin, Leather, Water, and Wood. The lower-left quadrant is composed of 32x32 blocks, and the lower-right quadrant of 16x16 blocks. The 128x128 blocks have been individually histogram equalized, the other blocks have been equalized by quadrant. Histogram equalization removes all first-order differences. It also finesses the problem of whether to measure image luminance or density, since the equalization gives the same result for either. We have also used a more rigorous adaptive equalization: it seems to give more conservative and reliable results.

The textures were chosen precisely because they are difficult to discriminate. They are a worst case dataset. Grass and Sand are very similar, with the main difference being the extended edges in grass. Pigskin, Raffia, and Sand may be considered cellular textures with similar cell sizes. Raffia is distinguished by its long-range structure, and Wool by its fiber content and lack of coarse edge structure. The Grass, Leather, Wood, and Water images all have vertical structure.

Comparison Statistics

Co-occurrence matrices are a popular source of texture features. We have generated co-occurrence matrices from 15x15 source windows requantized to 32 gray levels. Each matrix is thus 32x32. Nine of these matrices are used, with horizontal and vertical pixel spacings of zero, one, and seven pixels. The chosen spacings correspond to horizontal, vertical, and top-left to bottom-right diagonal directions. The P00 matrix records first-order information: all entries are on the diagonal. The other eight matrices record second-order information. The matrices are not symmetric, nor is there any averaging across different co-occurrence angles.

Table 1 shows classification accuracies available with various feature sets. The first analysis uses only the ASM, CON, COR, IDM, and ENT Haralick moments [1, 2]. Together the 32 features give almost 58% classification accuracy on the adaptively equalized texture set. The globally equalized textures generate two dominant discriminant functions using P10CON, P01IDM, P70IDM, P11CON, P01CON, P10IDM, P10COR, and P11COR. Discriminant functions for adaptively equalized textures use P10CON, P01IDM, P70CON, P11CON, P01CON, and P71COR. Angular second moment, correlation, and entropy features apparently carry little texture information.

The second and third analyses in Table 1 use rectilinear and diagonal moments, respectively. These moments will not be described here. Neither set is as powerful as the Haralick moments. The fourth analysis combines all of the co-occurrence features, a total of 172 independent texture measures for the nine co-occurrence matrices. Classification accuracy improves very little, and the variables selected by the discriminant analysis are nearly all from the Haralick set.

Macro-Statistic Selection

Figure 1 shows a one-to-one mapping between filtered images and texture energy planes. Twelve measures per pixel were used in preliminary research. Experience has shown that either variance or standard deviation alone is sufficient to extract texture information from the filtered images.

Variance is an average squared deviation from the mean. For a zero-mean field, it is an energy measure. The standard deviation (SDV) statistic is the square root of this local energy. It may be considered a "texture energy" measure. A faster energy transform is the average of absolute values (ABSAVE) within a window. For a zero-mean field it may be considered a fast approximation to the standard deviation. It performs poorly only with operators which are not zero-sum.

Table 2 shows classification accuracies possible with local texture energy measures. In each case the discriminant analysis routine selected about a dozen features, each produced by convolution with a 3-vector, 5-vector, 3x3 matrix, or 5x5 matrix. SDV and ABSAVE statistics were gathered over 15x15 "macro-windows." Classification accuracies are much higher than the 72% achieved with co-occurrence statistics. For this dataset, ABSAVE features are jointly more powerful than SDV features, and nearly as powerful as both sets together.

Rotation-invariant filters, such as the Sobel gradient magnitude, are only fair as texture measures. Better results are obtained by using directional masks separately and then combining the texture energy measures. Averages of ABSAVE values from rotated filter masks, such as L5E5 and E5L5, can be used as rotation-invariant texture measures. The usefulness of such measures depends on the application area. In a general vision system it is better to use directional measures, and to allow a higher level processor to decide which textures are equivalent.

Center-Weighted Filter Masks

Figure 3 shows two sets of one-dimensional convolution masks. The names are mnemonics for Level, Edge, Spot, Wave, and Ripple. The vectors in each set are ordered by sequency. The vectors are weighted toward the center, all are symmetric or antisymmetric, and all but the Level vectors are zero-sum. Vectors in each set are independent, but not orthogonal.

The 1x3 vectors form a basis for the larger vector sets. Each 1x5 vector may be generated by convolving two 1x3 vectors. S5, for instance, can be generated as (L3)*(S3), (S3)*(L3), or (E3)*(E3). A set of 1x7 vectors can be generated by convolving 1x3 and 1x5 vectors, or by twice convolving 1x3 vectors. The sequency of a generated vector is the sum of the component sequencies.

We have applied horizontal and vertical masks in pairs, although the discriminant analyses have not been constrained to assign equal weights. The six 3-vectors alone perform slightly better than the elaborate co-occurrence features. This is amazing considering the simplicity of the texture energy method and the many experimental vindications of Haralick's co-occurrence statistics. The 5-vector statistics perform even better, achieving 82% classification accuracy. Using 7-vectors or combining more than one vector size gives no significant improvement.

Neurological studies show that the visual cortex computes edge measures in approximately ten-degree increments. We have investigated one-dimensional features in the two main diagonal directions. Inclusion of diagonal features improves classification accuracies significantly. The 5-vector statistics alone are sufficient to achieve 86% classification accuracy, close to the maximum reached in this study. Combining different vector sizes adds little power, but provide insight into the feature selection process. The discriminant routine selects vectors of all directions and sizes. Different subsets are selected in the globally equalized and adaptively equalized cases. Yet all of the selected features are either Edge statistics or the symmetric statistics. None of the high-sequency antisymmetric features were found useful.

Figure 4 shows the nine masks generated by convolving a vertical 3-vector with a horizontal 3-vector. This may be considered a cross-product or vector multiplication operation, but convolution has special significance here. We shall extract texture information from image data by convolving with the 3x3 masks. Convolution with the component one-dimensional masks gives exactly the same result as convolution with a separable 3x3 mask.

The nine independent 3x3 masks form a complete set. Any 3x3 matrix can be expressed as a unique linear combination of the masks. The 5x5 masks and 7x7 masks also form complete sets, with even stronger weighting toward the center. The separable structure of these masks makes it feasible to apply them as spatial-domain filters. A 5x5 convolution, for instance, can be implemented as two 3x3 convolutions, a 5x1 and a 1x5 convolution, or two 3x1 and two 1x3 convolutions.

Two-dimensional masks are even more powerful than the tested sets of one-dimensional masks. Again the length five masks are best, although the evidence is less conclusive. Classification accuracies are in the range 86% to 88%. The adaptively equalized 3x3+5x5 feature subset differs from the 5x5 feature subset only by the inclusion of L3S3, the ninth and last feature to be added. Analyses with 7x7 masks have shown no significant improvement. Selected statistics again differ from one analysis to another, but high-sequency antisymmetric features are not useful. The consistent inclusion of R5R5 is somewhat surprising since matching image structures must be quite rare. This mask resembles a two-dimensional sinc or Bessel function. The similar S5S5 feature is individually very strong, but has little power when combined with other features.

Combining one-dimensional and two-dimensional features improves classification accuracy very little. Two-dimensional features enter the models first, followed by a few of the longer vector features. Again there are few Wave or antisymmetric features, despite the fact they are individually strong discriminating features. Otherwise the selection seems somewhat arbitrary.

Two sets of filter masks have been found which work well: rotated vector masks and separable square masks. Very likely the human visual system uses rotated circular masks. For digital image processing the square masks are the most convenient. Only the Level, Edge, Spot, and Ripple 5x5 masks are useful. Classification of 15x15 blocks can be done with accuracies above 86% using just the Level, Edge, and Ripple or the Level, Spot, and Ripple subsets.

Classification Results

The Level (or L5L5) texture energy transform is sensitive to changes in luminance level. Its moving-window average can be used as a brightness measure for segmentation purposes. Its standard deviation can be used as a local contrast measure. The other filters are inherently insensitive to low frequency luminance changes, and can be made invariant to contrast changes by taking the ratio of ABSAVE values to the L5L5 SDV values. This normalization reduces classification accuracy by about two percentage points. The contrast-invariant features may be used to segment or classify image textures without prior histogram equalization.

We have applied the contrast-invariant transforms to the composite texture image. Figure 2 shows the first two principle component planes before contrast normalization. The third plane (not shown) is similar to the second with contrast in the first quadrant reversed. The discriminant dimensions are the same ones found with co-occurrence features and with every other texture set we have tried.

Figure 2d shows the results of classifying every pixel into one of the eight texture categories. The 15 zero-sum texture transforms from the 5x5 Level, Edge, Spot, and Ripple subset were used. Processing time was about 30 minutes on a PDP KL/10. Using twelve or even nine features would produce similar results in less time.

It can be seen that the large blocks are almost perfectly classified. Average classification accuracy is near 87% for interior regions of the 128x128 blocks. The 32x32 blocks are well separated, and the 16x16 blocks are differentiated to an extent. We believe this perfomance to be unmatched by any other texture classifier or image segmentation system.

REFERENCES

1. R.M. Haralick, K. Shanmugam, and I. Dinstein, Textural Features for Image Classification," IEEE Trans. on Syst., Man, and Cybern., vol. SMC-3, pp. 610-621, Nov. 1973.

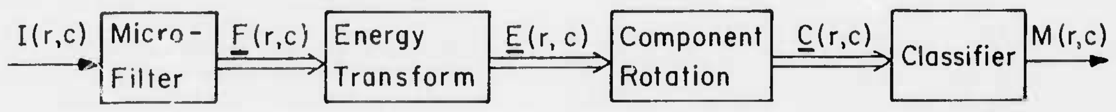
2. J.S. Weszka, C.R. Dyer and A. Rosenfeld, "A Comparative Study of Texture Measures for Terrain Classification," IEEE Trans. on Syst., Man, and Cybern., vol. SMC-6, pp. 269-285, April 1976.

Table 1. Co-occurrence Classification Accuracy.

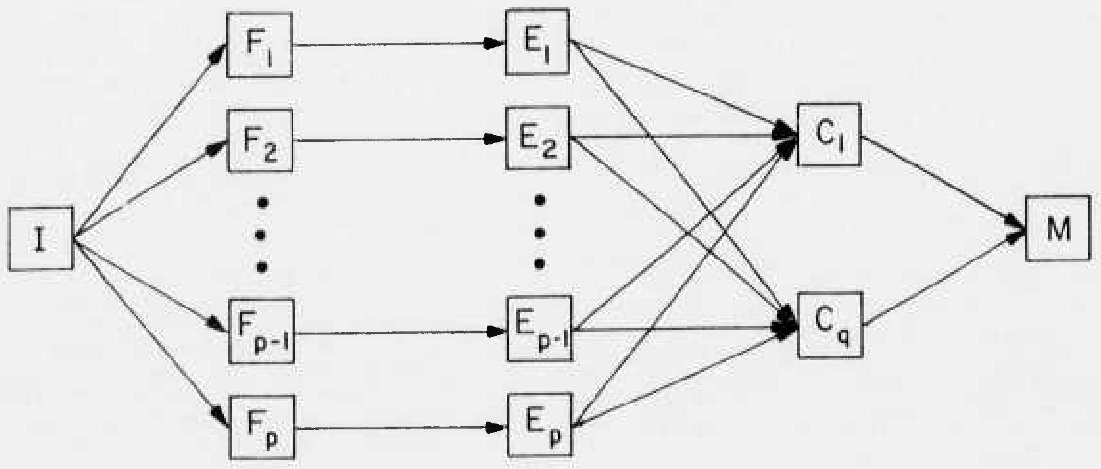
Feature Set	Global	Adaptive
Haralick Moments	70.85	67.58
Rectilinear Moments	63.04	65.92
Diagonal Moments	56.60	63.04
Combined Moments	72.07	68.16

Table 2. Macro-Statistic Classification Accuracies.

Feature Set	Global	Adaptive
SDV	85.99	85.60
ABSAVE	88.09	87.11
SDV+ABSAVE	89.16	87.55

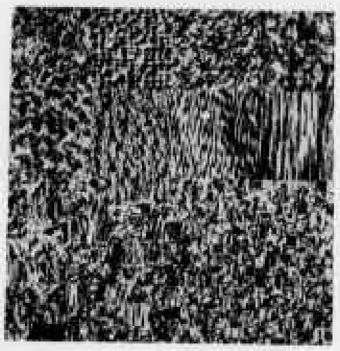


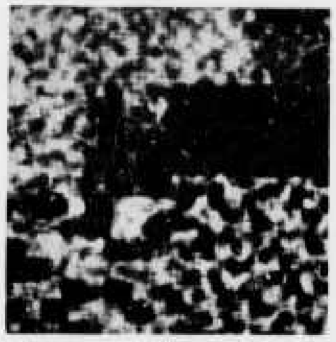
(a) Operator Sequence

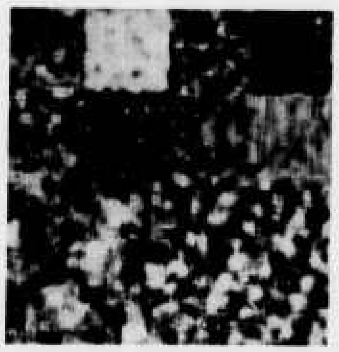


(b) Image Plane Sequence

Figure 1. Flow Diagrams.


(a) Composite


(b) First Component


(c) Second Component

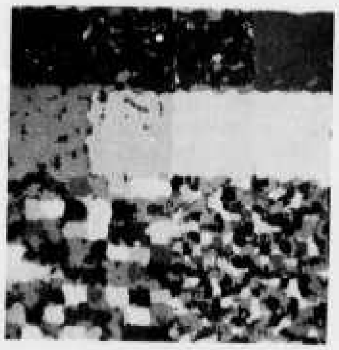

(d) 15x15 Classification

Figure 2. Texture Images.

```
L3 =  1   2   1
E3 = -1   0   1
S3 = -1   2  -1

L5 =  1   4   6   4   1
E5 = -1  -2   0   2   1
S5 = -1   0   2   0  -1
W5 = -1   2   0  -2   1
R5 =  1  -4   6  -4   1
```

Figure 3. Center-Weighted Vector Masks.

$$\begin{array}{ccc}
\begin{matrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{matrix} & \begin{matrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{matrix} & \begin{matrix} -1 & 2 & -1 \\ -2 & 4 & -2 \\ -1 & 2 & -1 \end{matrix} \\
\text{L3L3} & \text{L3E3} & \text{L3S3} \\[1em]
\begin{matrix} -1 & 2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{matrix} & \begin{matrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{matrix} & \begin{matrix} 1 & -2 & 1 \\ 0 & 0 & 0 \\ -1 & 2 & -1 \end{matrix} \\
\text{E3L3} & \text{E3E3} & \text{E3S3} \\[1em]
\begin{matrix} -1 & -2 & -1 \\ 2 & 4 & 2 \\ -1 & -2 & -1 \end{matrix} & \begin{matrix} 1 & 0 & -1 \\ -2 & 0 & 2 \\ 1 & 0 & -1 \end{matrix} & \begin{matrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{matrix} \\
\text{S3L3} & \text{S3E3} & \text{S3S3}
\end{array}$$

Figure 4. 3x3 Center-Weighted Masks

COOPERATIVE COMPUTATION IN TEXTURE ANALYSIS

Azriel Rosenfeld

Computer Vision Laboratory
Computer Science Center
University of Maryland
College Park, MD 20742

ABSTRACT

This paper describes several applications of simple cooperative computational techniques in texture analysis. The techniques involve parallel local operations of various sizes and resolutions performed on the given image. Interactions at a given size level, in the form of iterative local smoothing processes, can be used to improve the reliability of textural features measured at individual pixels or over windows. Interaction between operations of different sizes or resolutions can be used to check the textural homogeneity of windows or to detect and extract texture primitives.

1. INTRODUCTION

Texture plays an important role in the classification of terrain and land use types on aerial photographs and remote sensor imagery, particularly in the absence of multispectral information. For a recent review of texture analysis techniques see [1].

We will deal here primarily with the problem of classifying a given set of texture samples; in other words, we assume that we are given a set of image windows, each consisting of a single texture, and our task is to classify the windows into texture types, based on a set of feature measurements. A more difficult problem is that of segmenting a given image into uniformly textured regions.

A wide variety of statistical features can be used for texture classification. The following are three basic approaches:

a) One can use second-order gray level statistics, computed from "cooccurrence matrices" that represent the second-order probability density of gray level for various displacements. Alternatively, one can use first-order statistics of various local properties, e.g., of gray level differences for various displacements.

b) More generally, one can use second-order local property statistcs. These need not be computed for all pairs of points having a given displacement; rather, one can define a set of points that have characteristic local property values (e.g., local maxima of gray level difference), and compute second order local property statistics for pairs of these points having specific relationships (e.g., in the gradient direction) [2]. Alternatively, one can use local property values to select pairs of points in this way, and then compute second order gray level statistics for these pairs [3]. Each of these approaches has advantages in some situations.

c) One can segment the texture into "primitive elements", and compute first-order statistics of properties of these elements (area, perimeter, elongatedness, average gray level, etc.), or second-order statistics of properties of neighboring pairs of elements [4].

This paper describes several applications of simple cooperative computational techniques in texture analysis. The techniques involve parallel local operations of various sizes and resolutions performed on the given texture samples. In Section 2 we show how interactions at a given size level, in the form of iterative local smoothing processes, can be used to improve the reliability of texture features. We also discuss how this approach can be extended to include interactions between different sizes. Section 3 reviews some simple methods of extracting texture primitives, and shows how interactions over a range of sizes can be used to detect and extract such primitives.

2. TEXTURE FEATURE SMOOTHING

Typically, to obtain good classification performance, texture features should be computed for windows of size at least 64 by 64 pixels. If smaller windows are used, the feature values become unreliable, and classification performance deteriorates. However, the size of the windows required for reliable classification poses a problem when we try to analyze the textures on a given image; the larger the windows used, the fewer of them will fit inside the uniformly textured regions on the image, so that it becomes difficult to obtain sufficient numbers of uniformly textured samples.

This problem can be alleviated, to a substantial extent, by using small windows, and smoothing the texture feature values before using them for classification. The smoothing is done by selectively averaging the features measured for a given window with those measured for some of the neighboring windows; this should be done in such a way that the neighbors used are likely to have the same texture as the given window. A number of recently investigated image smoothing techniques should have the desired behavior; median filtering, and the E^k scheme in which we average with the k neighbors whose feature values are closest to those of the given window, are two examples.

A technical report on this method is in preparation; in this paper we give only one illustration. Figure 1 is a 512 by 512 pixel image composed of two geological terrain types (the dividing line is the 45° diagonal). Thus this image consists of 56 windows of size 64 by 64 that represent pure terrain types (28 of each), plus eight mixed windows lying on the diagonal. If we use windows of size 32 by 32, we have 120 pure windows of each type and 16 mixed windows; while if we use size 16 by 16, we have 496 pure windows of each type and 32 mixed windows. Note that the mixed area is reduced as the windows get smaller; the numbers of pixels belonging to mixed windows are 2^{15} for the 64 by 64's, 2^{14} for the 32 by 32's, and 2^{13} for the 16 by 16's.

A single second-order gray level statistic was used as the sole texture feature in this example. This feature was the moment of inertia of the co-occurrence matrix about its main diagonal (called the "Contrast" feature by Haralick); it was measured for a one-pixel displacement in the horizontal direction. The mean μ and standard deviation σ of the feature values for the windows of each size in each class are given in Table 1. These μ and σ values define a Gaussian probability density for each class and each size. We used these densities, in conjunction with Bayes' theorem, to compute the probability that each window belongs to each of the two classes, and to classify the window according to the greater of these probabilities. Since the densities overlap, particularly for the smaller windows, these maximum-likelihood classifications are not all correct; the error rates are shown in the first row ("iteration 0") of Table 2. (These rates refer only to the unmixed windows.)

The E^5 smoothing process was now iteratively applied to the feature values. Under this process, the variability of the values within each class rapidly decreases; this results in substantially reduced error rates when the windows are classified on the basis of their smoothed values, as Table 2 shows. No problems arise for windows near the border between the two textures, since the E^5 scheme is likely to average the features of such a window only with the features of neighboring windows of the same type. For the 64 by 64 windows, the error rate is reduced to almost zero in only a few iterations; while for the smaller windows, it is reduced to a level usually achievable only through the use of large windows.

In the experiments just described, only the features derived from windows of a single size were allowed to interact. Chen and Pavlidis [5] have described a split-and-merge method of texture segmentation in which feature values measured on windows of different sizes are compared. If the values for all four quadrants of a window are sufficiently close to the values for the entire window, then the window need not be subdivided; otherwise, we split it into quadrants. It would be very desirable to combine their method of "vertical" interaction between the feature values with our "horizontal" interaction method. Further work along these lines is planned.

Iterative smoothing can also be used to improve the results of image segmentation by pixel classification based on local property values. Such classifications are often somewhat noisy because the values are variable; for example, even in a "busy" region, local measures of "busyness" do not have uniformly high values. If we smooth the values, using a local smoothing scheme which tends not to cross region boundaries, the results are improved. Experiments along these lines are in progress.

As an alternative to iterative smoothing, one can use the initial feature values to probabilistically classify the windows (or pixels), and then use a relaxation scheme to adjust the class probabilities; this too should result in a reduced error rate. A comparison between the iterative smoothing and relaxation approaches is planned. It should be mentioned that in analogous studies of pixel classification based on spectral signatures, relaxation gave much better results than smoothing.

3. TEXTURE PRIMITIVE EXTRACTION

In [4], a region growing technique was used to extract texture primitives. Several simple methods of extracting primitives have also been investigated [6,7]. These included thresholding at a percentile, adaptive requantization (converting the window's histogram into a small set of spikes), and the SUPERSLICE algorithm; each of these schemes yields a set of connected components as primitives. The resulting set of components tends to be rather "noisy", i.e., some of them appear to be fragments of primitives, while others seem to be conglomerates of several primitives. Nevertheless, first- and second-order statistics (of area, elongatedness, etc.) computed from these components provide a useful basis for texture classification. Examples of the components obtained by these methods are shown in Figure 2.

A much "cleaner" set of primitives can be obtained using an edge-based approach [8], in which primitives are defined as clusters of antiparallel edge pairs. The basic idea is as follows: An edge detector is applied to the given texture window, and nonmaximum suppression, in conjunction with a low threshold, is used to select a set of edge points. At each such point, a search is made out to a fixed distance in the gradient direction. If an approximately antiparallel edge is encountered, the line segment joining the two edge points is assumed to be part of a primitive, and the points of this segment have their values incremented in an output array. When this has been done for all edge points, the points of the output array that belong to primitives should have high values. The final extraction of primitives is done by smoothing and thresholding the output array; see [8] for the details. Examples of the primitives obtained in this way are shown in Figure 3. Statistics derived from these primitves can then be used to classify the textures.

The edge-based approach to texture primitive classification can be implemented on several different levels. The implementation described in the preceding paragraph was based on individual edge points; for each such point, it searched the image in the appropriate direction for an antiparallel edge. A computationally cheaper idea might be to first link the edge points into edge segments, and then search the list of these segments to find antiparallel pairs.

One can also design implementations which do not require the explicit extraction of edge points from the given window, but rather operate on the raw gradient values. For example, suppose that for every point P we examine the set of pairs of points Q,R within a given distance of P that are symmetrically positioned with respect to P. (This involves a large number of operations for each P, but the process could be carried out quite efficiently using suitable parallel hardware.) Suppose that the gradient magnitudes at both Q and R are high, and the gradient directions are approximately antiparallel and roughly perpendicular to the line QR; then we increment the points of segment QR by an appropriate amount on an output array. The increment can be inhibited if some proper subsegment of QR has given rise to a higher increment. When this has been done for all triples P,Q,R, we should have high values in regions surrounded by antiparallel edges, and lower values elsewhere. Note that if we increment only P, rather than the entire segment QR, the results should be high on the "medial axes" of antiparallel edges, and low elsewhere; thus we have defined a generalization of the medial axis transformation to unsegmented gray scale images. This approach to primitive extraction and medial axis construction is currently under active investigation.

The conceptual advantage of this approach is that it requires no thresholding until the final stage, when a decision must be made as to which output array values represent primitives (or medial axes). The approach just described can be regarded as a process of cooperation and competition among a set of "dipole" operators of many lengths and orientations centered at each point; the dipoles having a given orientation compete, while those having different orientations cooperate.

Still another method of detecting and extracting texture primitives is to apply a set of spot (and streak) detectors, having a range of sizes (and orientations), to the given window. By a process of nonmaximum suppression with respect to position and size (and orientation), we can select a discrete set of locally best responses, which presumably correspond to primitives, assuming that the primitives are spot-like or streak-like. This process can be regarded as one of cooperation and competition among detectors of various sizes. The primitives that are detected in this way can then be extracted, if desired, by a local segmentation process, as described in a separate paper in these Proceedings. This approach was implemented many years ago in early studies of spot and streak detection using operators of multiple sizes; a one-dimensional version of it was recently used to define locally significant peaks in waveforms. Again, this is a computationally expensive approach, but it could be implemented very efficiently on suitable parallel hardware. Preliminary work along these lines is in progress, and further investigation of this approach is planned.

4. CONCLUDING REMARKS

The ideas sketched in this paper indicate how simple cooperative computational methods may have a variety of uses in texture analysis. The possibility of implementing such methods in parallel makes them potentially attractive for use in future real-time texture analysis systems.

REFERENCES

1. R. M. Haralick, Statistical and structural approaches to texture, Proc. IEEE 67, 1979, 786-804.

2. L. S. Davis, S. Johns, and J. K. Aggarwal, Texture analysis using generalized cooccurrence matrices, IEEETPAMI-1, 1979, 251-259.

3. C. R. Dyer, T.-H. Hong, and A. Rosenfeld, Texture classification using gray level cooccurrence based on edge maxima, Computer Science TR-738, University of Maryland, College Park, MD, March 1979.

4. J. T. Maleson, C. M. Brown, and J. A. Feldman, Understanding natural texture, Proceedings Image Understanding Workshop, Oct. 1977, 19-27.

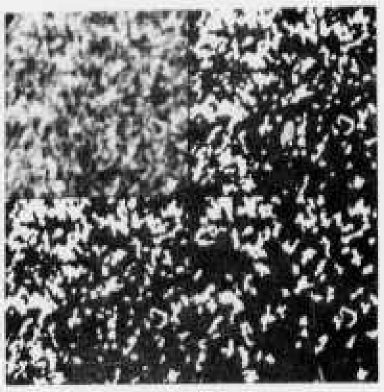
5. P. C. Chen and T. Pavlidis, Segmentation by texture using a co-occurrence matrix and a split-and-merge algorithm, CGIP 10, 1979, 172-182.

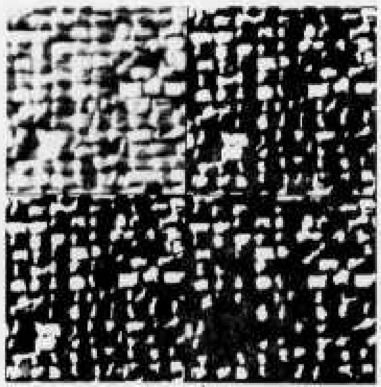
6. S. Wang, F. R. D. Velasco, and A. Rosenfeld, A comparison of some simple methods for extracting texture primitives and their effectiveness in texture classification, Computer Science TR-759, University of Maryland, College Park, MD, April 1979.

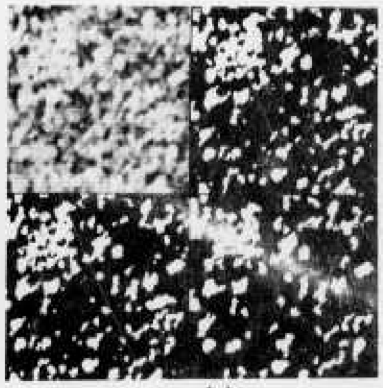
7. S. Wang, A. Wu, and A. Rosenfeld, Second-order statistics of texture primitives, Computer Science TR-779, University of Maryland, College Park, MD, July 1979.

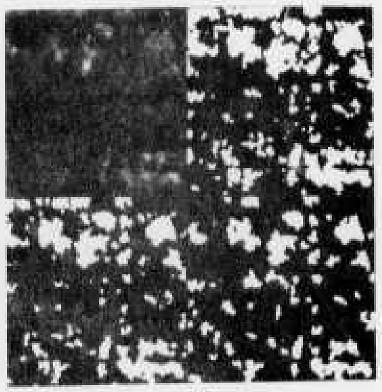
8. T.-H. Hong, C. R. Dyer, and A. Rosenfeld, Texture primitive extraction using an edge-based approach, Computer Science TR-763, University of Maryland, College Park, MD, May 1979.

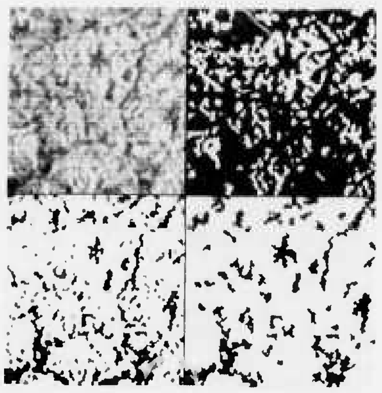
FIGURE 1 MAY BE FOUND ON THE FOLLOWING PAGE.

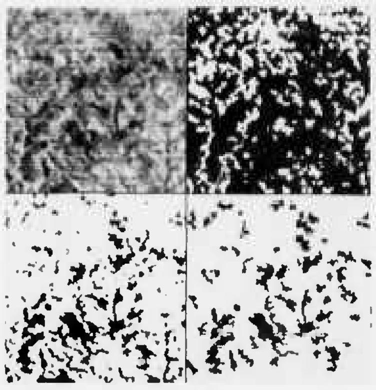

(a)


(b)


(c)


(d)


(e)


(f)

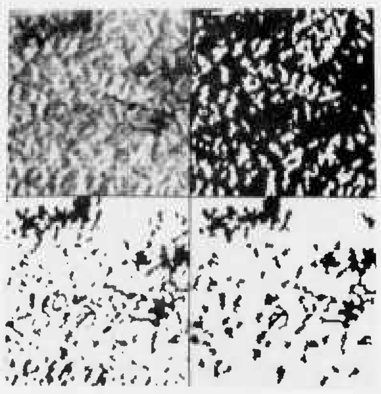

(g)

Figure 2. Primitive elements extracted from windows of seven textures using three simple methods. The textures are (a-d) grass, raffia, sand, and wool, from Brodatz's album; (e-g) Lower Pennsylvanian shale, Mississippian limestone and shale, and Pennsylvanian sandstone and shale from a LANDSAT image. The pictures in each part are as follows:

Original window	25th percentile	Original window	Darkest 25th percentile
Adaptive quantization	SUPERSLICE	Lightest 25th percentile	SUPERSLICE
Parts (a-d)		Parts (e-g)	

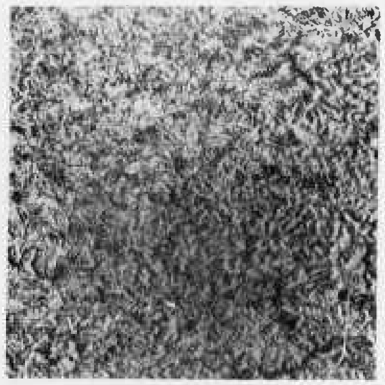


Figure 1. Geological terrain types selected from a LANDSAT frame. The upper left is Pennsylvania sandstone and shale; the lower right is Mississippian limestone and shale.

Terrain Type		Window Size		
		64 by 64	32 by 32	16 by 16
Pennsylvanian	μ	15.32	15.33	15.32
	σ	1.50	2.21	3.44
Mississippian	μ	11.66	11.67	11.68
	σ	1.20	1.75	2.65

Table 1. Means and standard deviations for the terrain types.

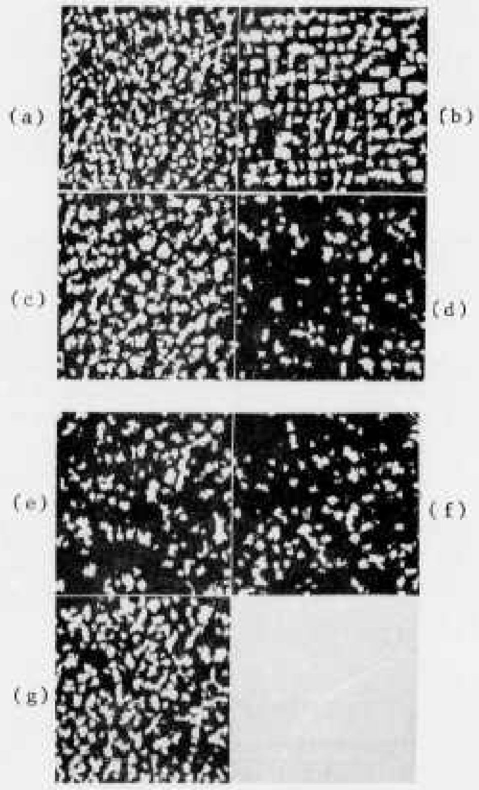


Figure 3. Primitive elements extracted from the same windows as in Figure 2 using the edge-based method.

Iteration	Window Size		
	64 by 64	32 by 32	16 by 16
0	9	19	28
1	2	8	19
2	2	5	16
3	2	4	14
4	2	4	14
5	2	3	13
6	2	3	13
7	2	3	13
8	2	3	12
9	2	3	12
10	2	3	11
11	2	3	11
12	2	2	11

Table 2. Error rates (%).

LIGHTS: A STUDY IN MOTION

Richard F. Rashid*
Department of Computer Science
University of Rochester
Rochester, NY 14627

*currently at Carnegie Mellon University
Pittsburgh, PA 15213

Summary

Lights is a system for the interpretation of simple moving light displays of jointed objects against a stationary background. The displays being studied differ from those examined by previous researchers in that (1) objects are represented by a relatively small number of points, (2) objects are not rigid, and (3) the viewing geometry is such that highly varying degrees of perspective distortion occur. An algorithm is presented which segments the points of an MLD of a wire-frame man into body parts. The relationship of this algorithm to previous theories of MLD perception and actual human performance is discussed.

The preparation of this paper was supported in part by the Defence Adavanced Projects Research Agency, monitored by the ONR under contract No. N00014-78-C-0164.

1. Introduction: Moving Light Displays

If asked what aspect of vision means the most to them, a watchmaker may answer "acuity," a night flier, "sensitivity," and an artist, "color." But to the animals which invented the vertebrate eye, and hold the patents on most of the features of the human model, the visual registration of *movement* was of the greatest importance. [1] (p. 342)

Motion supplies the visual system with crucial information about out environment. Indeed, motion information alone is sufficient for perception: A sequence of binary images representing points from a moving object can produce a strong and true-to-life three-dimensional perception.

Early in 1978 I set out to study just this kind of motion image, which I labelled a *moving light display* (MLD). I felt that MLD perception represented a severe challenge to existing notions about machine perception of multiple frame images.

An MLD isolates and presents geometric evidence of motion divorced from such factors as texture, color and lighting. The only source of information in an MLD is the position and velocity of its points, and position does not provide sufficient data for MLD interpretation. Psychological experiments have shown that individual frames of an MLD cannot usually be recognized by human subjects. [2]

So little information appears to be present in an MLD, that the question arises as to the nature of MLD perception: does the perception of MLDs require a large knowledge base to be used for hypothesis generation and model matching?, or do MLDs possess a structure which is exploited by the visual system as a shortcut to recognition?

2. Human performance

In looking for answers to these questions, it is instructive to consider just how good human beings are at interpreting MLDs. Johansson [3], for example, has demonstrated the that twelve moving lights can evoke the illusion of a walking man. His MLDs were created on video tape through the use of high intensity lights and adjustments of video contrast. Subjects performed a variety of tasks wearing glass bead reflectors on their major joints (shoulders, elbows, wrists, hips, knees and ankles), and the resulting MLDs of human body motion display considerable complexity. Less than .2 seconds were required for perfect recognition of an MLD as a moving man. Only .4 of a second was necessary for discrimination of different human movements, e.g. walking left, walking right, and walking backward.

The illusion of depth created by MLDs is very strong. When presented with a movie screen on which a small number of moving points are projected, a human observer will invariably try to place a three-dimensional interpretation on their movements. This is true even when abundant evidence of two dimensionality is present, such as the edge of the screen and the sound of the projector.

Human understanding of MLDs involves more than simple object identification and recognition. A considerable amount of information can be recovered from MLDs. Cutting has recently demonstrated the ability of subjects to recognize the sex of a walker [4], and it is even possible to recognize the gait of a friend [5].

3. Theories of MLD Interpretation

A number of theories have been developed to explain human perception of MLDs. I shall outline two of the most prominent, one from the field of psychology and one from computer science. A more complete critique of existing theories can be found in [6].

3.1 Johansson: Spatio-temporal integration

Johansson and his colleagues Borjesson and von Hofsten have attempted to explain the interpretation of MLDs in terms of a low level 'spatio-temporal differentiation and integration' [7]. The outer layers of the visual system, according to this theory, extract a hierarchy of coordinate systems that permit the interpretation of motion patterns according to a simple vector analysis.

In his 1976 paper Johansson describes the theory as it applies to the interpretation of the hip-knee-ankle system of an MLD of a man walking parallel to the viewing plane. The hip is identified as moving in the coordinate system of the stationary background. The knee moves in the coordinate system of the hip and the ankle in the coordinate system of the knee. Each point's total motion is seen as the composition of a movement relative to its particular coordinate system with the motion of that coordinate system relative to the next in the hierarchy.

Johansson suggests that the selection of a coordinate system for a point depends upon its two dimensional velocity. The lowest velocity point is interpreted relative to the stationary background and so on down the hierarchy. Unfortunately, this criterion does not always work even in his simple example. At certain points of the walker's step, e.g. when his foot is in contact with the floor, the movement of the ankle is actually less than the movement of either the hip or knee.

Despite their difficulty in defining rules for the determination of a coordinate hierarchy, Johansson et al. have presented a large body of data to corroborate their claim that the human visual system is performing a kind of vector decomposition in the analysis of MLDs. Their theory has led to the correct prediction of several MLD effects.

3.2 Ullman: The structure from motion theorem

A radically different approach has recently been suggested by Ullman [8]. He has demonstrated that three distinct orthogonal projections of four non-coplanar points provide sufficient information to reconstruct mathematically the three-dimensional structure of the object defined by the points (subject to a possible reflection). Using this 'structure from motion' theorem, Ullman has written a computer program capable of deriving the structure of multiple rigid objects in motion. He has also suggested an algorithm for the interpretation of MLDs of certain objects viewed by perspective transformation. He divides an object into rigid groups of four non-coplanar points, iteratively classifying overlapping groups of points in order to extract their relative three-dimensional location. The accuracy of this algorithm depends on the distances between the points selected in each step of the analysis. They must be close enough to each other (relative to the viewing distance) so that to a first approximation they are viewed by orthogonal projection.

Neither of these theories of MLD perception provides a basis for the interpretation of complex images. Ullman [8], for example, cannot cope with the low degree of connectivity, the perspective distortion, or the non-rigidity of MLDs such as those of human motion created by Johansson [3]. Johansson, on the other hand, presents only a partial solution, leaving out important details such as the determination of connectivity and coordinate bases.

4. Lights

I began my own study of MLDs by gathering extensive statistics on the position and velocity of MLD points. I hoped initially to demonstrate a strong mathematical relationship between the underlying objects and the movement of their 'lights', such as was shown to exist by Ullman for a restricted class of MLDs.

What I found was that strong relationships do exist between the movement of related points which are not dependent on a particular viewing transform (as in Ullman) and can not therefore be used directly for three-dimensional reconstruction. Instead, they derive from the fact that any perspective transform, even allowing for certain types of systematic distortion, tends over a period of time to preserve relationships between the movement of connected components of an MLD.

Lights is a computer system written to explore the ways in which this and other kinds of information can be exploited for the purpose of MLD interpretation. In its present form Lights is able to track and cluster points belonging to independently moving objects. Within a cluster, Lights analyzes the relative motions of object points. It then performs an initial segmentation of these points into groups representing independently moving subparts.

4.1 The Input to Lights

MLDs of human beings walking along different paths on a plane were chosen to be the primary stimuli for Lights. The reason for this choice was the high degree of difficulty represented by such images. The distance of each 'walking man' from the hypothetical viewer varies from about two to four times the man's height, creating an overall change in perspective distortion of 2.1. Typically, each man is seen to take about five steps in five seconds. Frames are displayed for about twenty-five milliseconds. The point of visual fixation remains constant (see Figure 4-1).

These MLDs were created by a program (written in SAIL) based on a model of human walking movement developed by Cutting [9]. Taken alone, the motions of the shoulders and hips define two ellipses having different major and minor axes. The arms and legs swing as double pendulums from the shoulders and hips and the entire body moves forward with each step. The speed of stride may be varied. As the speed is increased, a forward lean and accentuated arm and leg swinging are added. Other stimulus parameters include hip and shoulder excursion, speed, size, and three-dimensional path and orientation. The path of movement is defined either by a SAIL procedure which takes the current distance 'walked' and returns a three-dimensional coordinate or by a chain-coded path on a plane interactively specified on a screen (CRT) with a computer 'mouse'. The direction faced by the man is tangent to the path at all times.

Although referred to as a 'walking man', the underlying model is actually that of a wire-frame figure, since no attempt is made to occlude points on the basis of body part widths. Nevertheless, the net effect is a stimulus universally identified by human observers as a walking (albeit transparent) man.

Non-biological motions were studied using a program which simulated translation and rotation of geometric figures such as cylinders, squares, tetrahedrons, and less conventional objects such as 'jacks'. Once again, the underlying model was wire-frame and not solid so that no occlusion was possible.

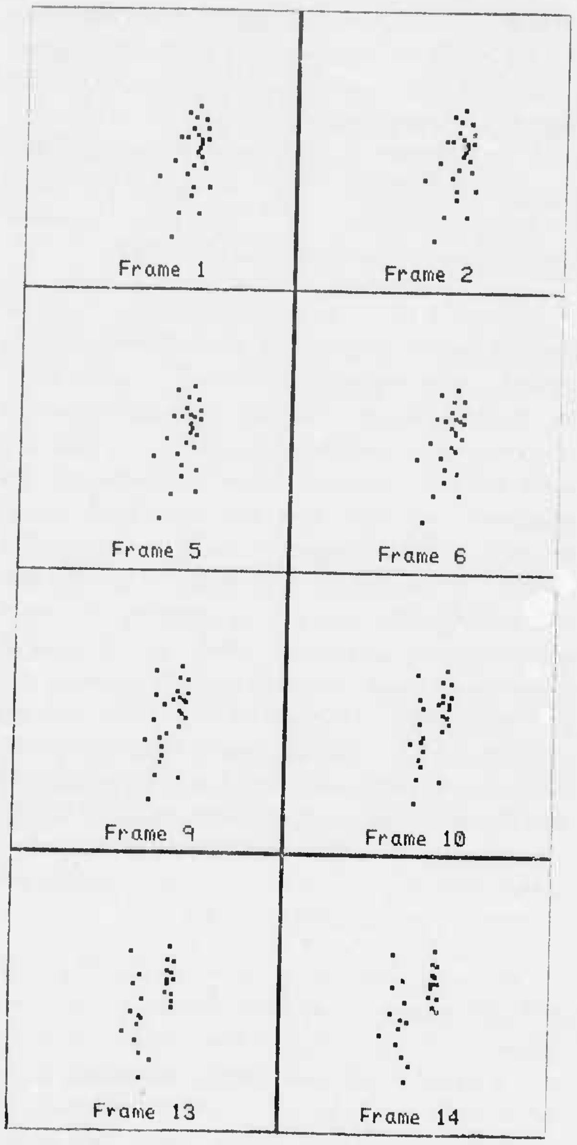


Figure 4-1: MLD of two men walking: representative frames

4.2 Components of MLD Interpretation

Lights breaks the problem of MLD interpretation into three components:

1. *Correspondence.* As presented to a viewer, an MLD contains no explicit data identifying points in one frame with points in another. This correspondence must be established before any further processing.

2. *Object separation.* Just as there is no explicit correspondence between points in successive frames, an MLD does not provide a ready-made solution to the problem of separating its points into groups which belong to different objects.

3. *Determination of subparts.* Once the points of an MLD have been divided into groups which are believed to correspond to distinct objects, it is necessary to break each object down into its component parts. This amounts to building a skeleton of the object by describing the connectivity relationships between its points.

4.3 Tracking

For each frame, the input to the interpretation program is an unlabeled set of coordinate pairs corresponding to the points of the MLD. The problem of tracking points from one frame to the next has been studied by others[10,8]. Often, though, tracking algorithms have been based on information (such as the cross correlation of small areas around prospective matches) derived from a greyscale image which served as the source for the MLD.

The MLDs under study here contain a small number of points and depict objects with parts in relative motion. The basic assumption is that the velocity of points in an MLD varies smoothly and can be used to estimate position from frame to frame.

4.3.1. The tracking algorithm.

The tracking algorithm used by Lights selects for each frame the correspondence which minimizes the sum of the differences between the expected position of each point (based on its velocity averaged over the preceding two frames) and the actual position of the corresponding point in the next frame.

Let m denote the number of points in frame F and n the number of points in frame F+1. Let P(F,i) represent the ith point in frame F for $1 \leq i \leq m$, and

P(F+1,j) represent the jth point in frame F+1 for $1 \leq j \leq n$. In addition, let Predict(F,i) be the function which takes the point P(F,i) in frame F and returns the predicted position of that point in frame F+1 based on its average velocity, and let $d^{F}_{F+1}(i,j)$ be the Euclidean distance between points Predict(F,i) and P(F+1,j). The correspondence desired is defined as the function $C_F(i)$ which maps indices for P(F,*) into indices for P(F+1,*) such that

$$\Sigma^{m}_{i=1} d^{F}_{F+1}(i, C_F(i))$$

is at a minimum.

It is important that the function $C_F(i)$ can be calculated efficiently. A naive approach would be to calculate all possible sums and choose the smallest, a feat requiring $O(m^n)$ operations.

Lights avoids this combinatorial explosion by applying a heuristic algorithm which will calculate $C_F(i)$ in $O(\mathrm{Max}(m,n)^2 \log(n))$ worst case time with an normal time of $O(\mathrm{Max}(m \log^2(n), n \log(n)))$. The idea for this algorithm came from the recognition of the fact that, in the images under study, the point selected by the function $C_F(i)$ was normallyu the point closest to Predict(F,i). This followed from the sparseness of the MLDs and the fact that the motions of their points corresponded to the motion of physical objects.

For each point in frame F the point closest to its predicted location in frame F+1 is calculated. This can be done in $O(\mathrm{Max}(m \log^2(n), n \log(n)))$ time using a Voronoi construction [11]. An array of n lists is then obtained with each list corresponding to a point P(F+1,j) in frame F+1 and containing the set of points in frame F for which P(F+1,j) is the best choice. This array is then traversed and lists with more than one element are examined. For list L(j) the best choice of element P(F,i) is made such that the sum of the distance from Predict(F,i) to P(F+1,j) and the distance between all other points in L and their next best selection in F+1 is at a minimum. All other points in the list are then distributed to the lists corresponding to their next best choice. For one pass of the array this algorithm requires at least $O(n \log(n))$ and at most $O(m\ n \log(n))$ time. The speed with which this calculation can be performed is due to the fact that the intermediate data structures used for constructing the original n-point Voronoi diagram can be reused to calculate an n-1 point Voronoi diagram. The new Voronoi diagram thus requires O(n) rather than $O(n \log(n))$ time to construct. Finding the next best match is then $O(\log(n))$ and a maximum of $(m-1) + 2 \times m$ additions are required to calculate the prospective sums. If there are no lists containing more than one element the algorithm has finished and calculated the function $C_F(i)$. In the majority (> 90%) of MLD frames studied, this condition occurs immediately and no iterations are required. Otherwise the algorithm is iterated a fixed number of times or until success is achieved.

If the tracking algorithm succeeds, the optimal match has been found. Optimality results from the fact that the selection of a match from a conflict list is always made in such a way that, if no further conflicts were to arise, the sum of all distances would be minimized. Once found, the list of point to point correspondences is then recorded and passed on to the later stages of Lights.

Failure of the heuristic does not imply failure of subsequent stages of the interpretation process. All failures are recorded, and an approximation to the best match is used in place of the optimal solution. Later stages of the system, however, treat the data from failure frames with caution.

4.3.2. Occlusion. The basic capabilities for dealing with occlusion were included into the tracking portion of the Lights system, even though no attempt was made to test the system with occluded MLDs. When, during the tracking process, a point appears for which there was no match in the previous frame or when the distance between an old point and its match is greater than three standard deviations from the mean, it is assumed that a new point has been added to the MLD. Points are assumed to have been deleted from the MLD when a suitable match can no longer be found. No attempt is made by the tracker to identify a point which has disappeared in the past with a newly discovered point. This function is more properly performed by later stages of the interpretation process based on object topology and world knowledge.

4.3.3. Experience with the tracking algorithm. The Lights tracking algorithm was devised and used to handle MLDs of common objects which have a high degree of predictability in their motions. In practice, the algorithm has worked extremely well. For MLDs derived from analytic functions (e.g. a man walking in a circle or straight line) perfect tracking is the rule. When the stimulus is generated by a chain-coded path, discontinuities of motion can cause tracking errors detectable only by later program stages. This is usually caused by the fact that the optimal match does not always correspond to the 'correct' match for such images. The algorithm has failed to find the optimal match in less than two percent of all frames examined. In all cases, tracking errors occur during frames which also cause difficulty for the human tracking system.

In one example, a roughly triangular path was drawn and the chain-code used for the construction of an MLD of a walking man. The best hand rounding of the triangle's corners still left them too sharp for smooth human turning motions, but the resulting display was considered very acceptable. When shown to a number of graduate students over the span of a few weeks, all reported seeing a 'normal' man walking along a triangular path with sharp turns. When the tracking program was run using this MLD as input, it mismatched the right knee with the left ankle after the first turn in the triangle. When these points were viewed in isolation, without the walking man to give them context, a number of people who had seen the previous display had the same impression of a switchover. Alerted to this illusion and re-shown the original MLD, all students saw the 'ankle turn into the knee' even though that was inconsistent with their interpretations of the rest of the display. This human tendency to ignore tracking errors unless they are explicitly pointed out suggested the strategy for handling such difficulties. When confronted with a possible conflict, the interpretation program simply suspends judgment on the identity of questionably matched points, waiting for a clear interpretation to present itself in later frames.

4.3.4. The use of velocity information. For some MLDs, particularly those with a single object, accurate tracking can be obtained without the use of a velocity estimate. This amounts to the assumption that no previous knowledge is necessary to map points from one frame to the next. It is normally the case, however, that information is known about the previous frames of an MLD. By using velocity information, more complex MLDs can be accurately tracked, even when wire-frame objects are seen to move in front of each other. Error rates were calculated for the Lights tracking algorithm for three different MLDs of increasing complexity. Three different values of the tracking algorithm's 'past history' parameter were used. With no past history taken into account (i.e. velocity averaged over zero frames) the simplest of the MLDs was nearly perfectly tracked, but the two more complex MLDs produced a large number of errors. As velocity was averaged over first one and then two frames more accurate correspondences were obtained.

The use of velocity information for tracking brings up the question of choosing initial conditions. Lights assumes that the correspondence between points in the first two frames of an MLD can be made on the basis of no past history (velocity). If a good match cannot be made,each new frame is examined in turn until this condition can be met.

4.4 Object Separation

Separation of MLD points into groups belonging to different objects is the next stage of Lights' interpretation process. The underlying assumption is that independently moving objects can be differentiated on the basis of their projected movement and position. When this assumption is violated, as in the case of two dancers arm in arm or soldiers marching side by side, the claim is that an MLD provides insufficient data to separate the objects. Higher level knowledge must be employed.

This approach to MLD interpretation departs from commonly held opinions in the field of motion research. Ullman [8] has criticized the grouping of elements into bodies as a prelude to structural analysis. He bases his stand on the fact that a Gestaltist grouping of points by 'common fate' is frequently inadequate for the separation of complex MLDs. Potter's criterion [12], for example, groups two points whenever their velocity difference falls below a defined threshold. Ullman cites the example of an MLD depicting two rotating cylinders one on top of the other as a demonstration of the problems with this technique. In such displays each cylinder contains points spanning a range of velocities and both may contain points moving at exactly the same speed.

The fact that simple rules for grouping points do not work should not be taken as sufficient grounds for abandoning the idea of low-level object grouping. Ullman was quick to give up object clustering because absolute structure determination was possible for his images. This solution is not available for the less restricted domain represented by MLDs of walking men.

Potter's less than satisfactory algorithm is, nevertheless, based on a reasonable assumption about the nature of velocity data from projected motion. Points in an image which correspond to the same moving object will exhibit, over time, relationships which can be exploited to separate them from other points in a scene. The problem with Potter's algorithm is that it does not take into account the fact that position as well as velocity is a key factor in determining the segmentation of a moving scene. Moreover, time is an important tool in motion understanding. It provides redundancy of information which can overcome errors and inadequacies in motion data. By utilizing all the information available in an MLD -- position, velocity and the redundancy of data in successive frames -- a way can be found around Ullman's objections in the techniques of graph-theoretic cluster analysis.

4.4.1. Clustering points into objects. Single-linkage cluster analysis has been successfully used to handle a wide range of problems such as separating two touching Gaussian distributions of points and determining gradient clustering [13]. It has been previously used in motion research to match segmented areas in successive frames of motion images [14]. This technique, based on the computation of the minimal spanning tree (MST), is used by Lights to distinguish independently moving objects.

Let every point in an MLD frame be represented by the four-vector (x, y, v_x, v_y), where x and y are its projected position and v_x and v_y its projected velocity (as determined by the tracking algorithm described in the previous section). A graph can be constructed which has each point as a node, with each node connected to all others by an edge of cost equal to their Euclidean distance. Information from previous frames is included by adding to this edge cost a function of the cost of the same edge in past frames. A minimal spanning tree can then be built [15] and the resulting graph can be segmented into clusters based on an appropriate cut function.

It is interesting to see how this algorithm functions on the example proposed by Ullman. Figure 4-2 shows the result of the algorithm on a frame of an MLD of two rotating cylinders viewed in orthogonal projection.

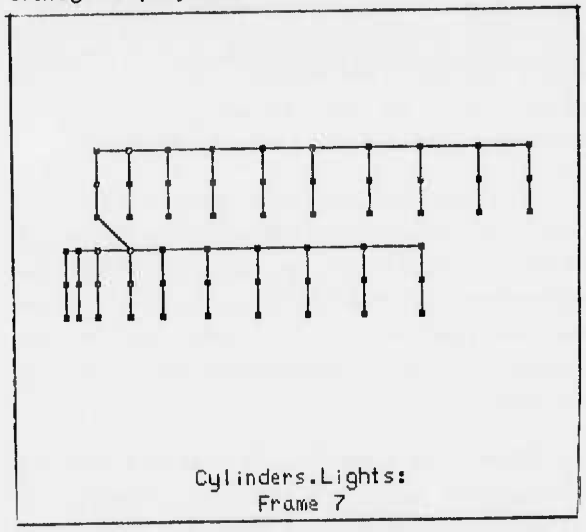


Figure 4-2: MST for two rotating cylinders

Thirty points were placed on each cylinder in such a way that no boundary could be seen in a static view of the first frame. After seven frames the MST for these points was calculated based on a cumulative distance function. While the projected velocity of points moving nearly parallel to the viewing plane did differ greatly from that of points moving nearly perpendicular to it, no sharp divisions occured within a cylinder because the speed of a point was close to that of its neighbors. On the border between the two cylinders, their different rotational velocities (four degrees per frame for the upper cylinder and two degrees per frame for the lower) resulted in a discontinuity which was found by the cluster analysis.

When a perspective rather than an orthogonal projection is used, changes in scale caused by varying degrees of perspective distortion can detract from the usefulness of data collected in previous frames. Lights compensates for these changes and for the mismatch in the units measuring velocity and position by scaling and translating each dimension of the four-dimensional feature space to have unit variance and zero mean. Single frame distances between features in this new space are combined with previous values to form a measure of the distance between points over a number of frames according to the function:

$$CD_n(i,j) = d(i,j) + CD_{n-1}(i,j) \times .95$$

where $CD_n(i,j)$ is the cumulative distance between points i and j in frame n and d(i,j) is the Euclidean distance between points i and j in frame n.

The criterion for separating clusters was conservatively chosen. Two clusters were assumed to be unrelated when the cost of the MST edge separating them was over fifty percent larger than the average cost of the edges near its two endpoints. A cluster was required to have at least two points.

Figure 4-3 shows the MST for two men, one walking in a circle, the other in a triangle, after thirty frames. Figure 4-4 below it shows another MST, this one calculated for two walking men traversing intersecting paths. In both cases a cut between the two groups of points could be made in twenty-five frames or less (about one-half step). Both examples were complicated by the fact that the projected positions of the two groups were initially close and by the fact that in both cases the men were made to walk 'in step' rather than show completely unrelated movement patterns. Greater

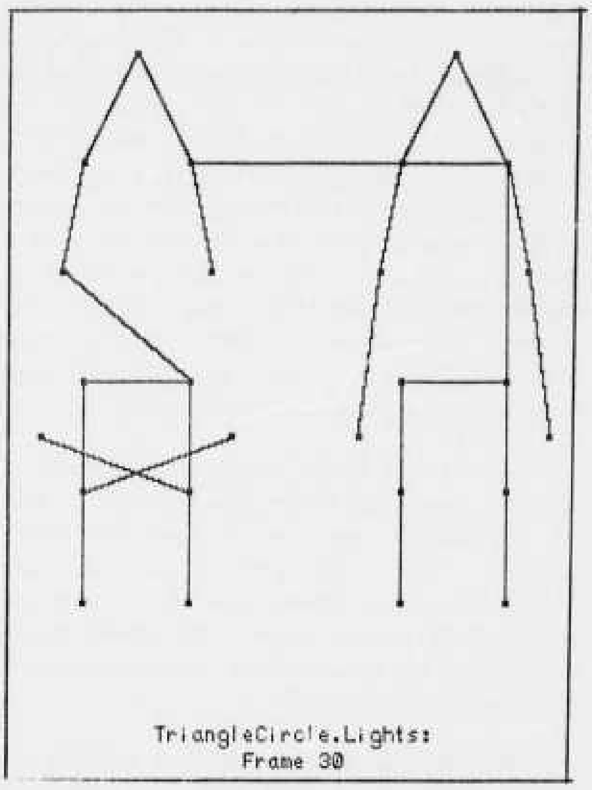


Figure 4-3: MST for two men after 30 frames

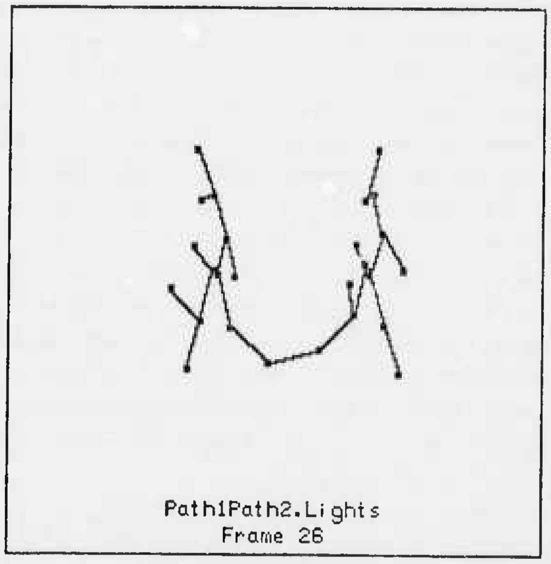


Figure 4-4: MST for two men after 26 frames

independence of movement would hasten the clustering process.

It should be noted that single-linkage clustering is but one of a group of clustering techniques including complete-linkage and average-linkage (King's method) clustering. Investigation is proceeding on the usefulness of these different clustering techniques and on the choice of cut criterion.

4.5 Intra-Object Relationships

An object in motion can be thought of as defining a moving coordinate system. Object parts move relative to that system and in turn define their own frames of reference. These two facts reflect not only the mechanics of motion but also its normal perception by a human observer.

Yet, particularly in the case of MLDs, this correspondence between object and percept seems singularly fortuitous. An infinite number of motions of points in space can produce a single MLD, and once a three-dimensional interpretation of structure is arrived at, it does not necessarily resolve such questions as 'what parts of an object are connected?' and 'how are unconnected parts related?'.

An informal experiment was devised to see how a group of graduate students and faculty members interpreted ambiguous connectivity information in MLDs. A display was constructed similar to the walking man displays discussed earlier but with the difference that the man remained rigid throughout his motion about a circular path (see Figure 4-5). The result corresponded roughly to a scene in which a mannequin is wheeled around in a circle or rotated on a lazy susan. Not only was the display understood as a rigid group of points moving through space, it was recognized immediately as a man in a fixed position. Other displays of rigid objects showed this same tendency to evoke a single perception of connectivity, despite the fact that all their points were equally 'connected' in the sense that an imaginary rod could be extended between them.

Certainly in the case of the rigid man moving in a circle, part of the explanation must lie in the sophisticated pattern matching abilities of the human mind. This may not, however, be the only reason. It may also be the case that the mechanisms used to interpret the structure of an object seen in an MLD are sensitive to certain relationships in the stimulus pattern, resulting in a tendency toward certain interpretations.

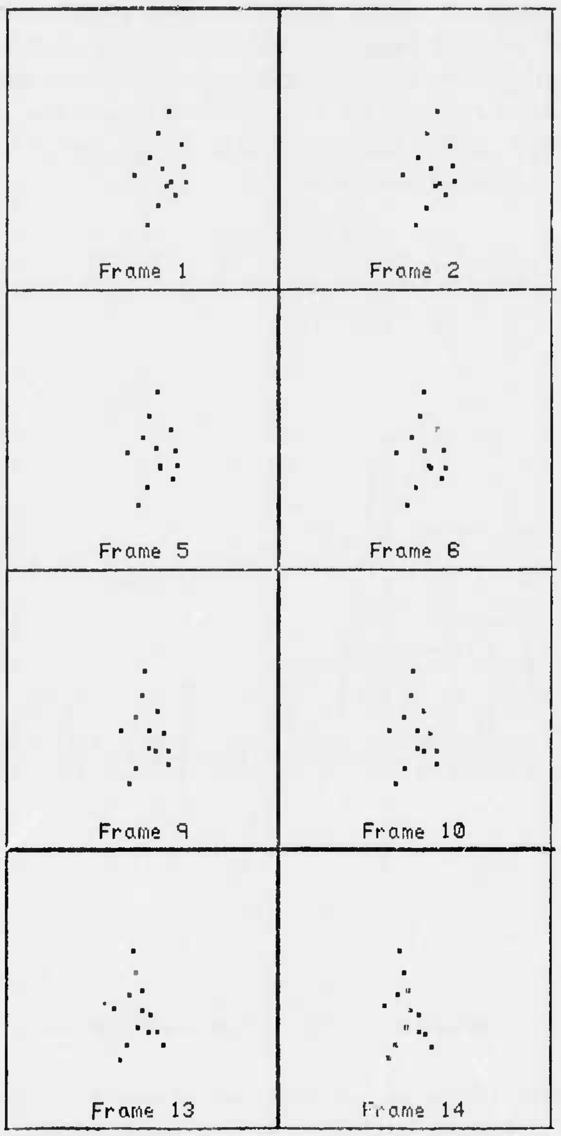


Figure 4-5: MLD of mannequin: representative frames

Whether or not this represents a credible theory of human vision, it is the case that the relationship calculation done by Lights on the points of an MLD (see previous section) can suggest connectivity in the underlying objects. Figure 4-6 shows the MST for three rigid objects - a man, a cube and a jack. There is a high degree of similarity between the connectivity preferred by most observers and the connections favored by the relationship function on which the MST was based.

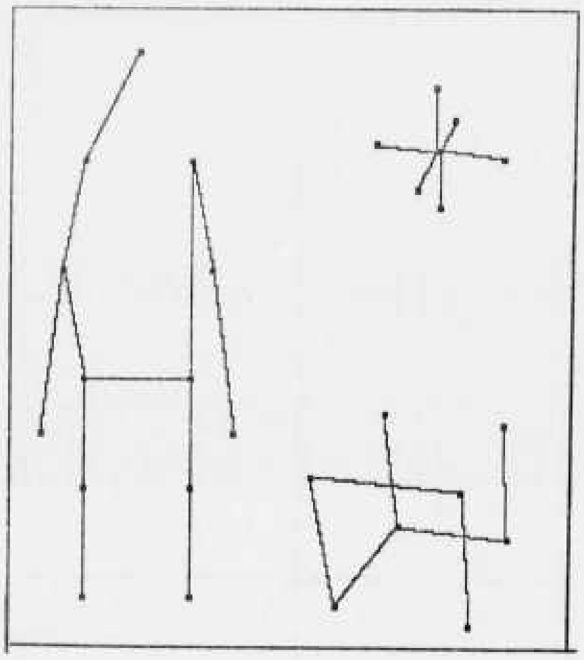


Figure 4-6: MST for three rigid objects

Initially it was hoped that this kind of clustering alone would lead to a natural breakdown of the object into subparts according to the following algorithm:

1. Separate individually moving objects using MST clustering.

2. Recalculate the MST for each object so defined.

3. Use this graph to define subparts.

Unfortunately, the groupings obtained from this algorithm did not always correspond to the correct division of the objects. The reason is that the clustering algorithm is meant to identify closely related points from their two-dimensional projection of position and velocity. For it to work properly, the relationships between the three-dimensional motions of connected points must be preserved. Often this will not happen if the object as a whole is spinning or twisting in space.

The calculation of similarity should most properly be done relative to the coordinate system defined by the moving object. Two facts define that system: (1) the movement of its origin, and (2) its changing orientation relative to the stationary background (the orientation itself is not important because there is no one 'correct' orientation for the object's coordinate system).

Lights attempts to compensate for these factors. The centroid of the points defining an object is used as an approximation for the origin of the object's frame of reference (psychologists have also used the centroid, see Borjesson and von Hofsten [16]). Some compensation for the rotation of an object is achieved through a modification to the Euclidean distance function to allow points with equal but opposite velocity to be considered 'close' together. The resulting MST more accurately reflects object composition. As can be seen in figures 4-7 and 4-8 the graph forms a singly-linked skeleton for the object.

The division of an object into its related parts is still subject to uncertainty. In the case of the walking man in particular, pseudo-relationships sometimes result from the similarity of motion of the arms and legs on opposite sides of the body. These graphs are useful nonetheless as a starting point for the next stages of the interpretation process - the recovery of three-dimensional relationships and the matching of the stimulus to a known model.

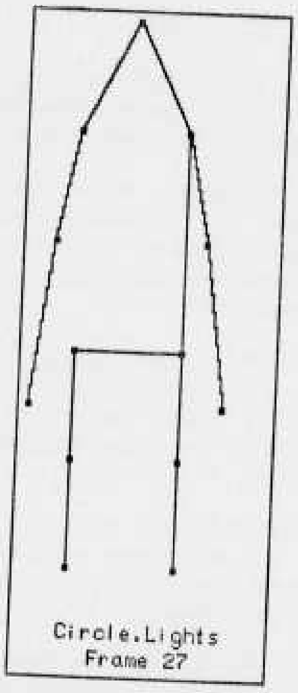


Figure 4-7: MST skeleton for walking man superimposed on canonical representation

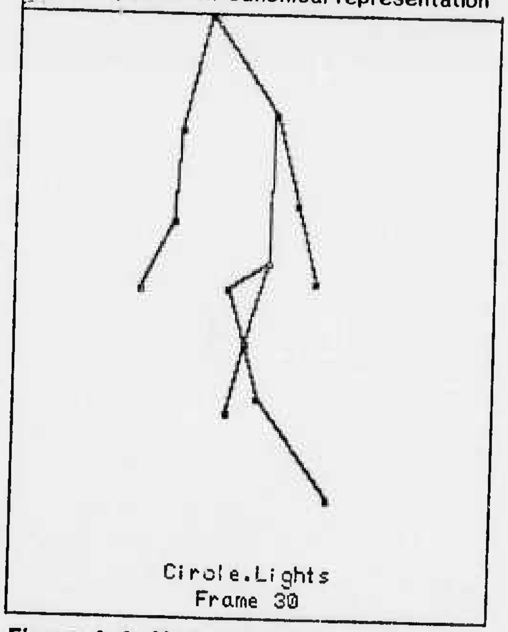


Figure 4-8: MST skeleton for walking man superimposed on actual frame from MLD

5. Conclusion

Lights demonstrates that MLDs possess an internal structure which can be a significant aid in the process of interpretation. This information is available independent of the kind of objects being observed and derives from the fact that, over time, related three-dimensional motions will exhibit relationships in their two-dimensional projection. These relationships hold true for both orthogonal and perspective projection and in the face of systematic distortions, and appear to explain the extraordinary robustness of MLD perception by human beings.

Lights has been used successfully on MLDs with one and two walking men and on images with geometric objects in motion. In addition, an MLD of a man walking a dog was recently constructed and was properly interpreted by Lights. Work is currently proceeding on the final phases of MLD interpretation: model-matching and description.

References

1. Walls, G. C., *The Vertebrate eye and its adaptive radiation*, Hafner, New York, 1963.

2. Johansson, Gunnar, "Visual motion perception," *Scientific American*, November 1976, pp. 76-88.

3. Johansson, Gunnar, "Visual perception of biological motion and a model for its analysis," *Perception and Psychophysics*, Vol. 14, No. 2, 1973, pp. 201-211.

4. Kozlowski, Lynn T. and Cutting, James E., "Recognizing the sex of a walker from dynamic point-light displays," *Perception and Psychophysics*, Vol. 21, No. 6, 1977, pp. 575-580.

5. Cutting, James E. and Kozlowski, Lynn T., "Recognizing friends by their walk: gait perception without familiarity cues," *Bulletin of the Psychonometric Society*, Vol. 9, No. 5, 1977, pp. 353-356.

6. Rashid, Richard, *Lights: A system for the interpretation of moving light displays*, PhD dissertation, University of Rochester, 1979, in preparation.

7. Johansson, Gunnar, "Spatio-temporal differentiation and integration in visual motion perception," *Psychological Research*, Vol. 38, 1976, pp. 379-393.

8. Ullman, Shimon, "The interpretation of structure from motion," 1978.Unpublished paper, Massachusetts Institute of Technology, Artificial Intelligence Laboratory.

9. Cutting, James E., "A program to generate synthetic walkers as dynamic point-light displays," *Behavior Research Methods and Instrumentation*, Vol. 10, No. 1, 1978, pp. 91-94.

10. Barnard, S. T. and Thompson, W. B., "Disparity analysis of images," Tech. report 79-1, University of Minnesota, January 1979.

11. Shamos, Michael, "Closest-point problems," *Proceedings of Sixteenth Annual Symposium on Fondations of Computer Science*, ACM, 1975, pp. 151-162.

12. Potter, J. L., *Extraction and utilization of motion in scene description*, PhD dissertation, University of Wisconsin, 1974.

13. Zahn, Charles T., "Graph-theoretical methods for detecting and describing Gestalt clusters," *IEEE Transactions on Computers*, Vol. C-20, No. 1, January 1971, pp. 68-86.

14. Dreschler, L. and Nagel, H.-H., "Using 'affinity' for extracting images of moving objects from TV-frame sequences," Bericht 44, Universitat Hamburg, Institut fur Informatik, February 1978.

15. Gower, J.C. and Ross, G.J.S., "Minimum spanning trees aand single linkage cluster analysis," *Applied Statistics*, Vol. 18, No. 1, 1969, pp. 54-64.

16. Borjesson, Erik and von Hofsten, Claes, "A vector model for perceived object rotation and translation in space," *Psychological Research*, Vol. 38, 1975, pp. 209-230.

MOTION DETECTION AND ANALYSIS

John Batali and Shimon Ullman

M.I.T. Artificial Intelligence Laboratory
545 Technology Square, Cambridge MA 02134, U.S.A.

Abstract

Motion information may be obtained from local measurements near the zero-crossing contours of image sequences convolved with a $\nabla^2 G$ mask. The information thus obtained may then be used to determine the boundaries and motions of objects in the image sequences. A computer implementation of the motion analysis is described.

Introduction

The extraction of motion information is an important stage in the early analysis of visual information. It can be subsequently used for a variety of useful tasks, e.g. the separation of moving objects from their background, and navigation by optical information.

In this paper we describe a technique for extracting motion information from a sequence of two (or more) images. We then outline a method for using the motion measurements in order to determine the boundaries of moving objects. These methods are based on the analysis of motion detection in [Marr & Ullman, 1979]. They have been recently implemented by John Batali, and tested on a number of both natural and computer-generated images.

Theoretical Overview

The analysis of motion in an image I is begun by convolving the image with a mask shaped like $\nabla^2 G$ where ∇ is the Laplacian operator and G is a symmetric 2-dimensional gaussian distribution:

$$G(x,y) = \frac{1}{2\pi\sigma^2} \exp \frac{-(x^2+y^2)}{2\sigma^2}$$

with σ the "space constant" of the gaussian.

Next, the zero-crossing contours (roughly speaking, the contours of zero value) in the filtered image are located. These zero-crossing contours serve as the basis for the computation of stereoscopic matching in a recent theory proposed by Marr and Poggio [1979], and for the computation of the primal sketch [Marr, 1976; Marr & Hildreth, 1979]. Zero-crossings correspond to loci of sharp intensity changes; they are useful in describing the changes that occur in an image at a particular scale. In addition, the representation of the filtered image by its zero-crossings is probably complete [Marr, Poggio & Ullman, 1979] and one can thus expect them to play an important role in early visual processing.

There are a number of reasons for basing the extraction of motion as well on the zero-crossing analysis. First, if zero-crossing contours form the basis of shape analysis, it is useful to determine their motion, thereby combining the analysis of shapes with the analysis of their motion. Second, the zero-crossings define contours along which the intensity gradient (perpendicular to the contours) is substantial, making it possible to obtain reliable velocity measurements. Third, the zero-crossings seem to be the earliest possible primitives for which reliable velocity can be obtained [Marr & Ullman, 1979].

However the use of zero-crossing contours raises a substantial difficulty if their velocity is to be measured by a local operator. This difficulty, which we shall term the *aperture problem*, is illustrated in figure 1. If the motion is to be detected by a unit that is small compared with the overall contour, the only information one can extract is the component of the motion locally perpendicular to the contour. Motion parallel to the contour would not be detected. Hence local measurements alone fail to give either the direction of the speed of movement, and can only restrict the direction to within $\pm 90°$. The use of zero-crossings (or other extended elements, such as edges and lines) decomposes the problem into two stages: the local measurements at the zero-crossing, and the subsequent combination of these measurements. We shall briefly review each in turn.

Measurements at the Zero-Crossings

Suppose we have available from a time-varying image: $I(x,y,t)$, its convolution with a $\nabla^2 G$ mask: $S(x,y,t) = \nabla^2 G * I(x,y,t)$, and the time derivative of the convolution: $T(x,y,t) = \frac{\partial}{\partial t} S(x,y,t) = \frac{\partial}{\partial t} \nabla^2 G * I$. From figure 2 it can be seen that if the zero-crossing is moving to the right, the value of the convolution at position Z will be increasing; and if the zero-crossing is moving to the left, the value will be

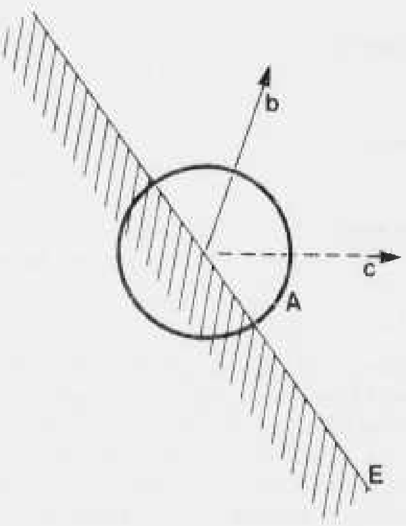


Figure 1. The Aperture Problem.

If the motion of an oriented element is detected by a unit that is small compared to the size of the moving element, the only information that can be extracted is the component of the motion perpendicular to the element. Looking at the moving edge E through a small aperture A, it is impossible to determine whether the actual motion is, e.g., in the direction of b or that of c.

decreasing. Hence, by examining the time derivative of the convolution at position Z, the magnitude of the motion relative to the orientation may be determined unambiguously. Figures 2b and 2c illustrate this.

In the case of motion to the right, when the zero-crossing reaches Z, T is strongly positive over a region centred on Z and $2\sigma_T$ wide, where σ_T is the space constant of the the gaussian used to obtain T. If motion is to the left, the sign of T is reversed.

In general, we have:

$$T = -v_r \hat{r} \cdot \nabla S$$

where $\hat{r}$ is a unit vector and v_r is the component of motion in the $\hat{r}$ direction. If we take $\hat{r}_s$ to be the direction perpendicular to the zero-crossing contour, then v_{r_s} may be determined by measuring $\hat{r}_s \cdot \nabla S$ and T.

In particular, the sign of the component of motion perpendicular to the zero-crossing segment is obtainable from the direction of $\hat{r}_s \cdot \nabla S$ and the sign of T. This is equivalent to an assertion about the direction of motion of the zero-crossing.

Combining the Measurements

Although the above measurements produce assertions that are only correct to within $\pm 90°$, by so doing they constrain the direction of motion to that range of values (figure 3a, b). The actual direction of motion may then be determined by combining the local constraints (figure 3c, d).

The general idea is to associate with each point on a zero-crossing contour an assertion about the possible motion at that point. A local operator then examines the assertions made in the vicinity of a point, and changes the assertion associated with the point to reflect the information obtained from the local region. The operators are considered to be working in parallel at each zero-crossing point in the image. As the assertions at various points in the image are modified, nearby operators will then use this modified information to further constrain the assertions. In this way, constraints "spread" throughout the image and the correct direction of the motions of whole objects may be determined.

Now, if desired, the actual angular velocity, $\hat{v}$ of an object may be calculated from $\hat{v} \cdot \hat{r}_s = v_{r_s}$. And discontinuities in either the directions or the velocities in the image indicate possible object boundaries.

Computer Implementation

Figure 4 shows the results of various stages of a computer implementation of the procedure. Figures 4a and 4b show the input images—two random-dot patterns. The squares are 512 by 512 pixels with a 50% density of 4 by 4 pixel dots. Figure 4b was constructed from figure 4a by moving a central square to the right and the background to the left.

When the two images are displayed to human subjects, with an interstimulus interval of about 10 msec, the central square is easily seen moving separately from the background.

The images are then convolved with a difference of gaussian mask whose central excitatory width is 6 pixels. An approximation to $\frac{\partial}{\partial t} \nabla^2 G * I$ is obtained by subtracting figure 4a from figure 4b and convolving the result with a mask of $w = 6$ pixels. This order of operation is justified because, due to the linearity of the derivative and convolution operations, we have:

$$\frac{\partial}{\partial t} \nabla^2 G * I = \nabla^2 G * \frac{\partial}{\partial t} I.$$

Figure 4c shows the result of the convolution of figure 4a, and figure 4d is the convolution of the difference image.

Zero-crossings are taken as the points in figure 4c whose values are nonnegative and where the sign changes from one side of the point to the other. These are shown in figure 4e.

The direction of $\hat{r}_s \cdot \nabla S$ is determined by fitting a line to a short segment of zero-crossings and recording the side of the segments where the values are positive. The local-motion direction is then found by examining the sign of the convolved difference image, T, at the corresponding point and applying the rule:

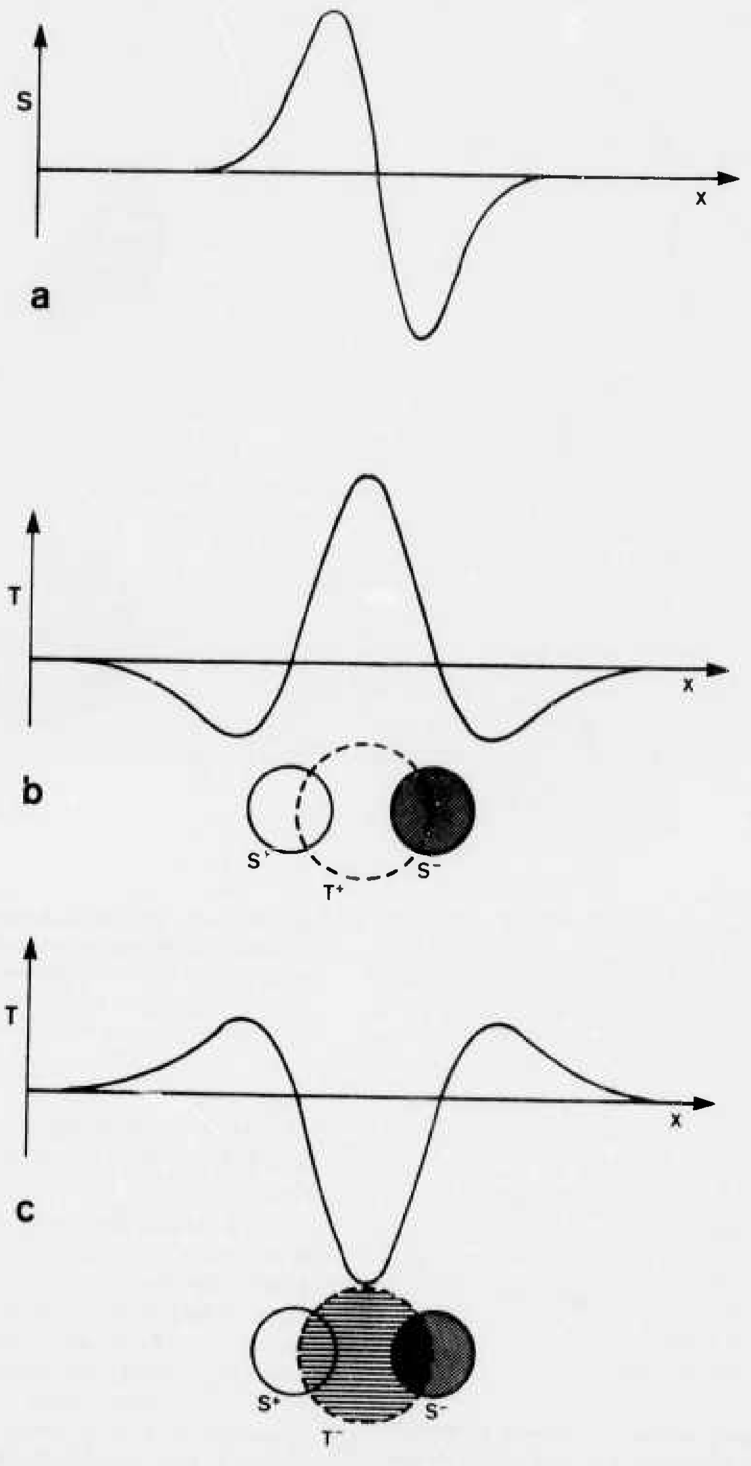


Figure 2. The Values of $S = \nabla^2 G * I$, and of $T = \frac{\partial}{\partial t} \nabla^2 G * I$ in the Vicinity of an Isolated Intensity Edge.

Figure 2*a* shows the S signal as a function of distance. The zero-crossing in the signal corresponds to the position of the edge. Figure 2*b* shows the spatial distribution of the T signal when the edge is moving to the right, and (2*c*) when it is moving to the left. Motion of the zero-crossing to the right can be detected when the S and T signals are as shown in figure 2*b*. Motion of the zero-crossing to the left is indicated in figure 2*c*

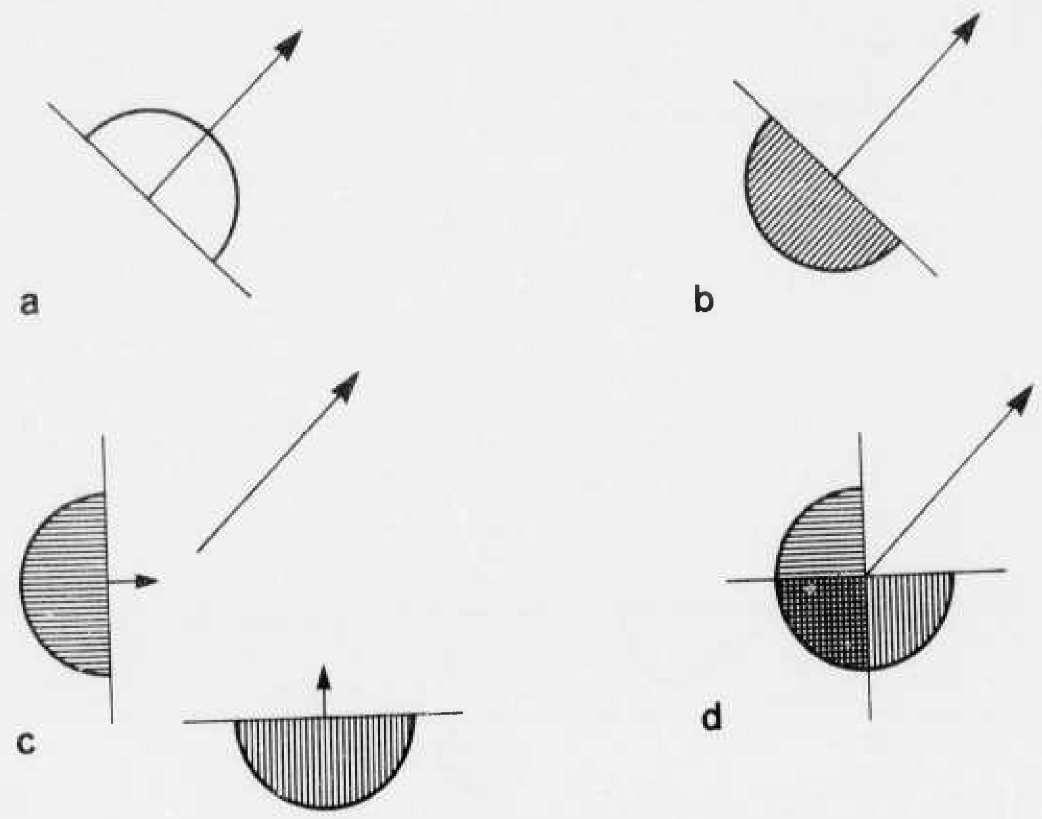


Figure 3. The Combination of Local Constraints.

The constraint placed by a single local-motion detector is that the direction of motion must lie within a range of 180° on the allowed side of the oriented element. (figure 3*a*). Equivalently, it is forbidden to lie on the other side, 3*b*. Figure 3*c* shows the forbidden zones for two oriented elements moving along the direction indicated by the arrow. The forbidden zone of their common motion is the union of their individual forbidden zones, as indicated in 3*d*. The direction of motion is now constrained to lie within the intersection of their allowed zones, i.e. the first quadrant.

(1) If the value of T is positive,
the motion is toward the negative side
of the zero-crossing contour.

(2) If the value of T is negative,
the motion is toward the positive side.

(3) If the value of T is zero,
the motion is ambiguous.

Figure 4*f* shows the points thus assigned a direction of motion from 315° to 45° (0° to the right). Figure 4*g* shows the points assigned a directions from 135° to 225°.

The combination of local constraints proceeds by modification of the motion assertions associated with each zero-crossing point. If motion was found at a zero-crossing point, we initially associate the set of allowed directions of motion with the point. Otherwise, we initially associate a no-motion assertion, as well as the direction of the positive side, with the zero-crossing.

A local operator, INT, then collects the assertions made in the vicinity of each zero-crossing point in the image and modifies the assertion at that point to reflect the information obtained in the local region. If the intersection of the sets of allowed directions associated with points in a small region is nonempty, INT modifies the assertion made at the central point of the region to allow only the directions in the intersection. If the entire image were of a single, rigid object in pure translation, and the local-motion detectors were perfect, each operation of INT on a local region would further constrain the allowed direction of motion around the correct direction. Continued iterations would eventually be equivalent to a "global" intersection converging on a single allowed direction—the actual direction the object moved.

If the intersection of the allowed directions in a local region around a point is empty, it means one of two things: Either one of the measurements in the region was in error, or the region contains an occluding edge between two objects moving in different directions. We call such points "nil-consistent".

Nil-consistent points are important for two reasons: First, if they occur at an occluding contour, they represent the desired output of the separation process. A nil-consistent contour is a very strong indicator of an object boundary. Secondly, we don't want to attempt to find the intersection of consistent directions over a region that gave rise to a nil-consistent point. If produced by an error, it is possible that the region contains other errors and it would be good to minimize their effect. If the nil-consistent point was produced by an edge, we wish to intersect only within the object's boundaries—not across them.

So when the local region being examined by an INT operator contains a nil-consistent point, the operator makes no changes in the region. For moving objects, we thus intersect o[illegible] within the object boundaries and avoid error points. I[illegible]e object contains zero-crossing points at many orientations, the allowed direction of motion will be tightly constrained after a few INT iterations.

If no motion was found at a zero-crossing point (i.e. the value of T was zero), we could have one of two situations: Either there was actually no motion at the point; or the motion at the point was parallel with the zero-crossing contour. To account for these situations, the INT operator treats all **no-motion** assertions as consistent if no other type of assertion is found in the region. If all no-motion assertions in the region are associated with zero-crossing segments with the same orientation, and the intersection of the set of allowed directions associated with zero-crossings where motion was found includes one of the directions parallel with the non-moving contours, that direction will be the assertion INT attaches to the central point of the region. In all other cases, non-motion and motion assertions are taken to be inconsistent, and **nil-consistent** assertions are associated with points in regions that contain both. The addition of these considerations allows the separation of a moving object from a non-moving background as well as the inclusion of the very strong constraints suggested by movement parallel with one of the object's contours. If we allow "tracking" of one of two objects moving in the same direction but at different speeds, then that object can be seen as the "ground" for the the other object and thus the two can be separated.

The INT operator collected the assertions made in a 12 by 12 pixel square region around each point and modified the assertion at the point in the appropriate way. The nil-consistent points found after one iteration are shown in fig *4h*.

All of the computations in both the motion-detection and combination of constraint programs are local in the sense that they only use values in a few nearby points. In biological systems this sort of spatially limited locality is important because it allows the processing to be done very quickly, in parallel over the entire image. Information need not be transmitted very far, so no long interconnections are needed.

Both of these considerations apply also to the implementation of the computations in VLSI hardware. Current design work taking advantage of this locality is being done on a VLSI implementation of an oriented zero-crossing detector.

References

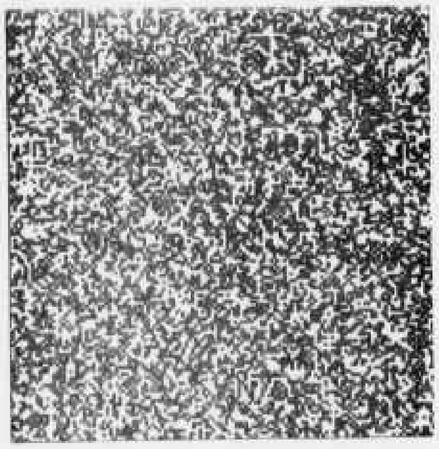
D. Marr, "Early Processing of Visual Information," *Phil. Trans R. Soc. Lond. B*, 275, 483–524 1976

D. Marr and E. Hildreth, "Theory of Edge Detection," *Proc. R. Soc. Lond. B*, 1979 (submitted for publication.)

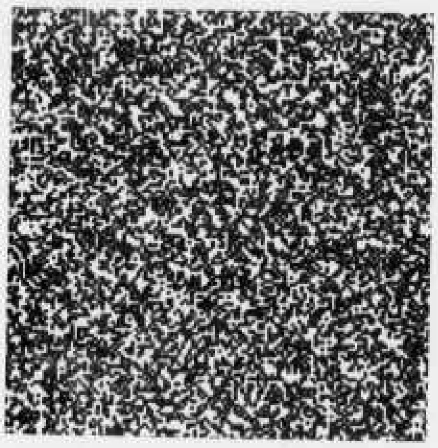
D. Marr and T. Poggio, "A Computational Theory of Human Stereo Vision" *Proc. R. Soc. Lond. B*, 204, 301–328 1979

D. Marr and S. Ullman, "Directional Selectivity and its Use in Early Visual Processing" *A.I. Memo 524*, M.I.T. Artificial Intelligence Laboratory, 1979.

D. Marr, S. Ullman and T. Poggio, "Bandpass Channels, Zero-Crossings and Early Visual Information Processing," *J. Opt. Soc. Am.*, **69:6** 914–918 1979.



A



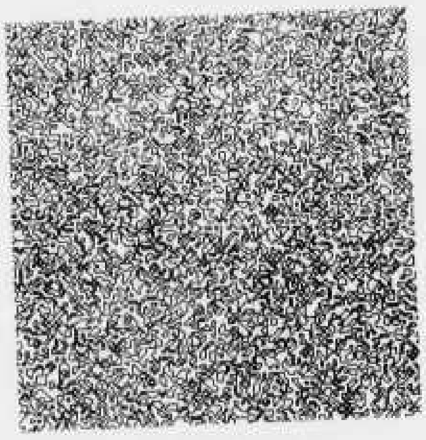
B

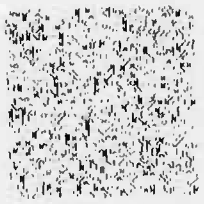
C

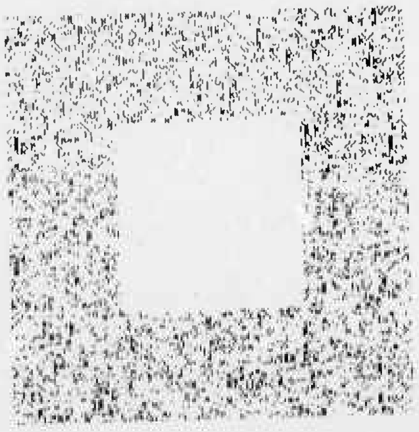
D

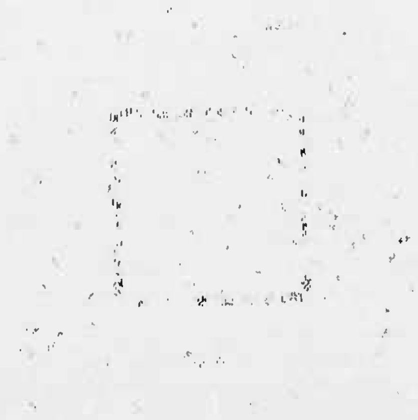
Figure 4. The Motion of Random Dots

Figure 4*a* shows a 512 × 512 pixel random-dot pattern with a 50% density of 4 × 4 pixel dots. In 4*b* the centre of 4*a* is shifted one pixel to the right, the background is shifted one pixel to the left. Figure 4*c* shows the convolution of the first image with a difference of gaussian mask whose $w = 6$ pixels. The convolution of the image created by subtracting 4*a* from 4*b* is shown in 4*d*. The zero-crossings of the first image are shown in figure 4*e*. 4*f* shows the points where the local-motion algorithm found motion to the right, and 4*g* shows zero-crossings that moved to the left. Figure 4*h* shows the nil-consistent points found after one iteration of the **INT** operator.


E


F


G


H

RECONSTRUCTING SMOOTH SURFACES FROM PARTIAL, NOISY INFORMATION

H. G. Barrow and J. M. Tenenbaum

SRI International
Menlo Park, California

ABSTRACT

Interpolating smooth surfaces from boundary conditions is a ubiquitous problem in early visual processing. We describe a solution for an important special case: the interpolation of surfaces that are locally spherical or cylindrical from initial orientation values and constraints on orientation. The approach exploits an observation that components of the unit normal vary linearly on surfaces of uniform curvature, which permits implementation using local parallel processes. Experiments on spherical and cylindrical test cases have produced essentially exact reconstructions, even when boundary values were extremely sparse or only partially constrained. Results on other test cases seem in reasonable agreement with human perception.

INTRODUCTION

Surface perception plays a fundamental role in early visual processing, both in humans and machines [1, 2]. An explicit representation of surface structure is directly necessary for many low-level visual functions involved in applications such as terrain modeling, navigation, and obstacle avoidance. It is also a prerequisite for general-purpose, high-performance vision systems.

Information about surfaces comes from various sources: stereopsis, motion parallax, texture gradient, shading, and contour shape, to name a few. Information may be provided in terms of absolute or relative values of orientation or range, depending upon the nature of the source. Moreover, different techniques for extracting this information are valid in different parts of the scene. For example, inferring shape from shading is difficult on a highly textured surface, or in areas of complex illumination, while stereo information is not available in textureless areas nor areas visible only from one viewpoint. Thus, in general, evidence is incomplete, may be quite sparse (as in line drawings), and subject to noise, which leads to ambiguity.

Any attempt to derive globally consistent surface descriptions from these diverse local sources must therefore address the following basic computational problems:

(1) interpolation of sparse data

(2) smoothing of noisy data

(3) deciding which techniques are applicable in which parts of the scene

(4) integration of different types of data from different sources

(5) deciding the location and physical type of boundaries

In this paper we look mainly at the first problem, which arises in virtually all theories of low-level vision [1, 2]. We principally address the problem of reconstructing a smooth surface, given a set of initial orientation values, which may be sparse or only partially constrained.

COMPUTATIONAL PRINCIPLES

We begin with a precise definition of the reconstruction problem in terms of input and output.

The input is assumed to be in the form of sparse arrays, containing local estimates of surface range and orientation, in a viewer-centered coordinate frame. In practice, the estimates may be clustered where the information is obtainable, such as along curves corresponding to surface boundaries. In general, they are subject to error and may be only partially constrained. For example, given a three-dimensional boundary, the surface normals are only constrained to be orthogonal to the boundary elements. We also assume that the location and nature of all surface boundaries are known, since they give rise to discontinuities of range or orientation. This last condition is required in the current implementation and is intended to be relaxed at a later date to accommodate imperfect boundary detection.

The desired output is simply filled arrays of range and surface orientation representing the most likely surfaces consistent with the input data. Refinement of hypothesized surface discontinuities is also desired. These output arrays are analogous to our intrinsic images [1] or Marr's 2.5D sketch [2].

For any given set of input data, an infinitude of possible surfaces can be found to fit arbitrarily well. Which of these is best depends upon assumptions about the nature of surfaces in the world and the image formation process. Ad hoc smoothing and interpolation schemes which are not rooted in these assumptions lead to incorrect results in simple cases. For example, given a few points on the surface of a sphere, iterative local averaging [3, 4] of range values will not recover a spherical surface.

Assumptions about Surfaces

The principal assumption we make about physical surfaces is that range and orientation are continuous over them. We further assume that each point on the surface is essentially indistinguishable from neighboring points. Thus, in the absence of evidence to the contrary, it follows that local surface characteristics must vary as smoothly as possible and that the total variation is minimal over the surface. Range and orientation are both defined with reference to a viewer-centered coordinate system, and so they cannot directly be the criteria for evaluating the intrinsic smoothness of hypothetical surfaces. The simplest appropriate measures involve the rate of change of orientation over the surface; principal curvatures (k1, k2), Gaussian (total) curvature (k1*k2), mean curvature (k1+k2), and variations upon them all reflect this rate of change [5]. Two reasonable definitions of smoothness of a surface are uniformity of some appropriate measure of curvature [6], or minimality of integrated squared curvature [7]. Uniformity can be defined as minimal variance or minimal integrated magnitude of gradient.

The choice of a measure and how to employ it (e.g., minimize the measure or its derivative) depends, in general, upon the nature of the process that gave rise to the surface. For example, surfaces formed by elastic membranes (e.g., soap films) are constrained to minimum energy configurations characterized by minimum area and zero mean curvature [8]; surfaces formed by bending sheets of inelastic material (e.g., paper or sheet metal) are characterized by zero Gaussian curvature [9]; surfaces formed by many machining operations (e.g., planes, cylinders, and spheres) have constant principal curvatures.

We are not prepared, at this point, to maintain that any of these measures is inherently superior, particularly because of various close relationships that exist between them. We note, for example, that minimizing the integrated square of mean curvature is equivalent to minimizing the sum of integrated squares of principal curvatures and the integrated Gaussian curvature, G, as shown by:

$$\int (k1 + k2)^2 .da = \int k1^2 .da + \int k2^2 .da + 2\int k1*k2.da \qquad (1)$$
$$= \int k1^2 .da + \int k2^2 .da + 2\int G.da$$

We also note that making curvature uniform by minimizing its variance of any measure over a surface is equivalent to minimizing total squared curvature, if the integral of curvature is constant. This follows from the well-known fact that for any function, f(x),

$$\text{Variance of f} = \int (f-fbar)^2 .dx \qquad (2)$$
$$= \int f^2 .dx - \left[\int f.dx\right]^2 / DX$$

On any developable surface for which Gaussian curvature, G, is everywhere zero, and on a surface for which orientation is known everywhere at its boundary (e.g., the boundary is extremal), the integral of G is constant. Thus, for such surfaces, minimizing variance of G and minimizing its integrated square are equivalent.

By itself, however, uniformity of Gaussian curvature is not sufficiently constraining. Any developable surface is perfectly uniform by this criterion, so considerable ambiguity remains, as is evident in Figure 1, where all of the developable surfaces satisfy the same boundary conditions. Thus a secondary constraint, such as uniformity of mean curvature, is required to find the smoothest developable surface.

In this paper we focus on surfaces with reasonably uniform curvature--surfaces that are locally spherical or cylindrical. We shall demand exact reconstructions for spherical and cylindrical test cases and intuitively reasonable reconstructions for other smooth surfaces. In particular, given surface orientations defined around a circular outline, corresponding to the extremal boundary of a sphere, or along two parallel lines, corresponding to the extremal boundary of a right circular cylinder, we require interpolation to yield the correct spherical or cylindrical surface, with uniform (Gaussian, mean, and principal) curvature. These cases are important because they require reconstructions that are symmetric in three dimensions and independent of viewpoint. Many simple interpolation techniques fail this test, producing surfaces that are too flat or too peaked. Given good performance on the test cases, we can expect reasonable performance in general.

A RECONSTRUCTION ALGORITHM

Although in principle correct reconstruction for our test cases can be obtained in many ways, the complexity of the interpolation process depends critically upon the representation. For example, representing surface orientation in terms of gradient space leads to difficulties because gradient varies very nonlinearly across the image of a smooth surface, becoming infinite at extremal boundaries. We shall now propose an approach that leads to elegantly simple interpolation for our test cases.

Coordinate Frames

Given an image plane, we shall assume a right-handed Cartesian coordinate system with x- and y-axes lying in the plane (see Figure 2). We also assume orthogonal projection in the direction of the z-axis. Each image point (x,y) has an associated range, Z(x,y); the corresponding scene point is thus specified by

$$(x, y, Z(x,y)) \quad .$$

Each image point also has an associated unit vector that specifies the local surface orientation at the corresponding scene point:

$$N(x,y) = (Nx(x,y), Ny(x,y), Nz(x,y)) \quad .$$

Since N is normal to the surface Z,

$$\begin{aligned} Nx/Nz &= - dZ/dx \\ \text{and} \quad Ny/Nz &= - dZ/dy \quad . \end{aligned} \tag{3}$$

(The derivatives dZ/dx and dZ/dy correspond to p and q when the surface normal is represented in gradient space form, (p,q,-1).)

Differentiating equation (3), we obtain

$$\begin{aligned} d(Nx/Nz)/dy &= - d^2 Z/dy.dx \\ \text{and} \quad d(Ny/Nz)/dx &= - d^2 Z/dx.dy \quad . \end{aligned} \tag{4}$$

For a smooth surface, the terms on the right of (4) are equal, hence

$$d(Nx/Nz)/dy = d(Ny/Nz)/dx \quad . \tag{5}$$

Finally, since N is a unit vector,

$$Nx^2 + Ny^2 + Nz^2 = 1 \quad . \tag{6}$$

Semicircle

Let us begin by considering a two-dimensional version of surface reconstruction. In Figure 3 observe that the unit normal to a semicircular surface cross section is everywhere aligned with the radius. It therefore follows that triangles OPQ and PST are similar, and so

$$OP : OQ : QP = PS : PT : TS \quad . \tag{7}$$

But vector OP is the radius vector (x,z) and PS is the unit normal vector (Nx,Nz). Moreover, the length OP is constant (equal to R) and the length PS is also constant (equal to unity). Hence,

$$Nx = x/R \quad \text{and} \quad Nz = z/R \quad . \tag{8}$$

Sphere

Now consider a three-dimensional spherical surface, as shown in Figure 4. Again the radius and normal vectors are aligned, and so from similar figures we have

$$Nx = x/R \quad Ny = y/R \quad \text{and} \quad Nz = z/R \quad . \tag{9}$$

The point to note is that Nx and Ny are both linear functions of x and y, and that Nz can readily be derived from Nx and Ny because vector N has unit length.

Cylinder

The case of the right circular cylinder is only a little more complex. In Figure 5 observe a cylinder of radius R centered upon a line in the x-y plane, inclined at an angle A to the x axis. Let d be the distance of point (x,y,0) from the axis of the cylinder. Then

$$d = y.\text{Cos } A - x.\text{Sin } A \tag{10}$$

$$\text{and} \quad z^2 = R^2 - d^2 \quad . \tag{11}$$

Let Nd be the component of vector N parallel to the x-y plane; it is clearly perpendicular to the axis of the cylinder. Now, since a cross section of the cylinder is analogous to our first, two-dimensional, case,

$$Nd = d/R \quad . \tag{12}$$

Taking components of Nd parallel to the x and y axes,

$$Nx = Nd.\text{Sin } A \quad \text{and} \quad Ny = -Nd.\text{Cos } A \quad . \tag{13}$$

Substituting in this equation for Nd, and then for d,

$$Nx = (y.\mathrm{Cos}\ A - x.\mathrm{Sin}\ A).\mathrm{Sin}\ A/R \qquad (14)$$

and

$$Ny = -(y.\mathrm{Cos}\ A - x.\mathrm{Sin}\ A).\mathrm{Cos}\ A/R \ .$$

Observe that as for the sphere, Nx and Ny are linear functions of x and y, and that Nz can be derived from Nx and Ny.

INTERPOLATING SPHERICAL AND CYLINDRICAL SURFACES

From the preceding section, we can see that to interpolate values for the normal vector, on spherical and cylindrical surfaces, between points where its value is known, we need only determine the linear functions that describe the components Nx and Ny. This can be done simply from known values at any three noncollinear points. The resulting functions can be used to predict precisely values of Nx and Ny, and hence Nz also, over the entire surface. The vector field produced is guaranteed to satisfy the integrability constraint of Equation 5, as may be verified by substituting for Nx, Ny, and Nz from Equations 9 or 14 (for the sphere or cylinder, respectively) and 6. Finally, the orientation field can be integrated to recover range values.

For the special test cases, because of the global nature of the linearity of Nx and Ny, it is possible to interpolate between given boundary values, treating Nx and Ny as essentially independent variables. While in general the integrability constraint should not be ignored, in practice, since complex surfaces can often be approximated locally by spheres or cylinders, this constraint is weak and its omission does not result in significant errors.

A COMPUTATIONAL MODEL

We have implemented a model that uses parallel local operations to derive the orientation and range over a surface from boundary values. It exploits the linearity and separability results for the test cases and extends them to arbitrary smooth surfaces.

The overall system organization is a subset of the array stack architecture first proposed in [1]. It consists conceptually of two primary arrays, one for range and the other for surface normal vectors, which are in registration with each other (and with the input image). Values at each point within an array are constrained by local processes that maintain smoothness and by processes that operate between arrays to maintain the differential/integral relationship. In general, we must be able to insert initial boundary values sparsely in both range and orientation arrays and have the system relax to fill in consistent intervening values. At present we know how to handle the restricted case where only orientation is initially specified.

THE INTERPOLATION PROCESS

At each point in the orientation array we can imagine a process that is attempting to make the two observable components of the normal, Nx and Ny, each vary as linearly as possible. The process looks at the values of Nx (or Ny) in a small patch surrounding the point and attempts to infer the linear function, f = ax + by + c, that best models Nx locally. It then tries to relax the value for the point to reduce the supposed error.

There are numerous ways to implement such a process, and we shall describe some of the ones with which we have experimented. One of the simplest is to perform a local least-squares fit, deriving the three parameters a, b, and c. The function f is then used to estimate a corrected value for the central point. The least-squares fitting process is equivalent to taking weighted averages of the values in the patch, using three different sets of weights:

$$\sum_i x_i Nx_i\ , \quad \sum_i y_i Nx_i\ , \quad \sum_i Nx_i \ . \qquad (15)$$

The three parameters of f are given by three linear combinations of these three averages.

If we are careful to use a symmetric patch with its origin at the point in question, the sets of weights and the linear combinations are particularly simple--the three sums in equation (15) correspond, respectively, to

$$a*\sum_i x_i^2\ , \quad b*\sum_i y_i^2\ , \quad c*\sum_i 1 \ . \qquad (16)$$

Equations (15) and (16) can be readily solved for a, b, and c; but note that under the above assumptions, f(0,0)=c, so computation of a and b is unnecessary for updating the central point, unless derivatives are also of interest.

An alternative approach follows from the fact that a linear function satisfies the equation

$$\nabla^2 f = 0 \ . \qquad (17)$$

Numerical solution of this equation, subject to boundary conditions, is well known. The ∇^2 operator may be discretely approximated by the operator

$$\begin{array}{ccc} & -1 & \\ -1 & 4 & -1 \\ & -1 & \end{array} \ .$$

Applying this operator at a point in the image leads to an equation of the form

$$4Nx_0 - Nx_1 - Nx_2 - Nx_3 - Nx_4 = 0 \quad , \qquad (18)$$

and hence, rewriting,

$$Nx_0 = (Nx_1 + Nx_2 + Nx_3 + Nx_4)/4 \quad . \qquad (19)$$

Equation (19) is used in a relaxation process that iteratively replaces the value of Nx_0 at each point by the average of its neighbors. Although the underlying theory is different from least-squares fitting, the two methods lead to essentially the same discrete numerical implementation.

The iterative local averaging approach works well in the interior regions of a surface, but difficulties arise near surface boundaries where orientation is permitted to be discontinuous. Care must be taken to ensure that the patch under consideration does not fall across the boundary, otherwise estimation of the parameters will be in error. On the other hand, it is necessary to be able to estimate values right up to the boundary, which may, for example, result from another surface occluding the one which we are attempting to reconstruct.

The least-squares method is applicable to any shape of patch, which we can simply truncate at the boundary. However, the linear combination used to compute each parameter depends upon the particular shape, so we must either precompute the coefficients for all possible patches (256 for a 3x3 area) or resort to inverting a 3x3 matrix to derive them for each particular patch. Neither of these is attractive.

The above disadvantages can be overcome by decomposing the two-dimensional fitting process into several one-dimensional fits. We do this by considering a set of line segments passing through the central point, as shown in Figure 6. Along each line we fit a function, $f = ax + c$, to the data values, and thus determine a corrected value for the point. The independent estimates produced from the set of line segments can then be averaged. If the line segments are each symmetric about the central point, then the corrected central value is again simply the average of the values along the line. The principal advantage of the decomposition is that we can discard line segments which overlap a boundary, and often at least one is left to provide a corrected value. We would prefer to use short symmetric line segments, since they form a compact operator, but in order to get into corners we need also to resort to one-sided segments (which effectively extrapolate the central value). We have implemented a scheme that uses the compact symmetric operator when it can, and an asymmetric operator when this is not possible (see Figure 7).

We have experimented with a rather different technique for coping with boundary discontinuities, which is of interest because it involves multiple interrelated arrays of information. For each component of the orientation vector we introduce two auxiliary arrays containing estimates of its gradient in the x and y directions. For surfaces of uniform curvature, such as the sphere and cylinder, these gradients will be constant over the surface; and for others, we assume they will be slowly varying. To reconstruct the components of the normal, we first compute its derivatives, then locally average the derivatives, and finally reintegrate them to obtain updated orientation estimates.

Derivatives at a point are estimated by considering line segments through the point parallel to the axes. We again fit a linear function--but now we record its slope, rather than its intercept, and insert it in the appropriate gradient array. In the interior of a region we may use a symmetric line segment, and near boundaries, a one-sided segment, as before. The gradient arrays are smoothed by an operator that forms a weighted average over a patch, which may easily be truncated at a boundary. (To form the average over an arbitrarily-shaped patch, it is only necessary to compute the sum of weighted values of points within the patch and the sum of the weights, and then divide the former by the latter.) A corrected orientation value can be computed from a neighboring value by adding (or subtracting) the appropriate gradient. Each neighboring point not separated by a boundary produces such an estimate, and all the estimates are averaged.

ESTIMATION OF SURFACE RANGE

The process of integrating orientation values to obtain estimates of range Z is very similar to that used in reintegrating orientation gradients. We again use a relaxation technique, and iteratively compute estimates for Z from neighboring values and the local surface orientation. Here we need orientation expressed as dZ/dx and dZ/dy, which are obtained from Nx and Ny by Equation 3. At least one absolute value of Z must be provided to serve as a constant of integration. Providing more than one initial Z value constrains the surface to pass through the specified points; but since the inverse path from Z to N has not yet been implemented, the resulting range surface is not guaranteed to be consistent with the orientations.

EXPERIMENTAL RESULTS

An interactive system was implemented in MAINSAIL [10] to experiment with and evaluate the various interpolation algorithms discussed above. This system includes facilities for generating quadric surface test cases, selecting interpolation options, and plotting error distributions.

Test Cases

How well do each of the above interpolation techniques reconstruct the test surfaces? To answer this, we performed a series of experiments in which the correct values of Nx and Ny were fixed along the extremal boundaries of a sphere or cylinder, as shown in Figure 8. The surface orientations reconstructed from these boundary conditions were compared with those of ideal spherical or cylindrical surfaces generated analytically.

The first set of experiments involved a sphere of radius 7 centered in a 17 x 17 interpolation array. We deliberately used a coarse grid to test the accuracy of the reconstruction under difficult conditions. (A coarse grid also has the experimental advantage of minimizing the number of iterations needed for convergence.) Correct values for Nx and Ny were fixed at points in the array falling just inside the circular extremal boundary of the sphere. Table I summarizes the results for this test case, using various interpolation operators.

The results on the spherical test case are almost uniformly good. In all cases, except gradient smoothing, the maximum absolute error is below one percent after 100 iterations ($-1.0 < Nx, Ny < 1.0$). On any cross section through the sphere, the maximum error occurs approximately a quarter of the way in from both boundary points, the error being zero at the boundary points and also on the symmetry axis half way between them. We conclude that 8-connected, uniformly weighted averaging and 8-way linear interpolation/extrapolation are superior in terms of speed of convergence, with the linear operator preferred because of its advantages at boundaries and corners. These conclusions generalize to all of the test cases we have studied to date. Thus, for brevity, the experimental results that follow are reported only for the 8-way linear operator.

The second set of experiments involved a cylinder of radius 6, centered in an 8 x 8 interpolation array. Again, correct values for Nx and Ny were fixed at points in the array falling just inside the parallel lines representing the extremal boundaries of the cylinder. With the cylinder oriented parallel to the X or Y axis, the maximum absolute error in Nx or Ny after 50 iterations was .018 and the RMS average error .01. After 100 iterations, the absolute error dropped to .0004 and the RMS average to .0002. When the major axis of the cylinder was inclined 60 degrees to the X-axis, the errors look much higher: .12 absolute and .03 RMS after 50 iterations; .108 absolute and .03 RMS after 100 iterations; .09 absolute and .02 RMS after 300 iterations. However, the errorful orientations were concentrated solely in the upper right and lower left corners of the array, where the cylinder boundary is effectively occluded by the array edge. Extrapolation of values from the central region, where the orientations are very accurate, into these partially occluded corners accounts for the slow rate of convergence. After 1,000 iterations, however, orientations are highly accurate throughout the array.

Other Smooth Surfaces

Given that orientations for uniformly curved surfaces can be accurately reconstructed, the obvious next question is how well the algorithms perform on other surfaces for which curvature is not globally uniform. A simple case to consider is that of an elliptical boundary. However, we immediately run into the problem of what is to be taken as the "correct" reconstruction. When people are asked what solid surface they perceive, they usually report either an elongated object or a squat object, roughly corresponding to a solid of revolution about the major or minor axis, respectively. The elongated object is preferred, and one can argue that it is more plausible on the grounds of general viewpoint (a fat, squat object looks elongated only from a narrow range of viewpoints). When presented with initial orientations for an elliptical extremal boundary (Figure 9), our algorithms reconstruct an elongated object, with approximately uniform curvature about the major axis. They, in effect, reconstruct a generalized cylinder [11], but without explicitly invoking processes to find the axis of symmetry or matching the opposite boundaries.

In a representative experiment, initial values for Nx and Ny were fixed inside an elliptic-shaped extremal boundary (major axis 15, minor axis 5). The reconstructed orientations were then compared with the orientations of the solid of revolution generated when the ellipse is rotated about its major axis. The resulting errors after 50 iterations were: for Nx, .02 maximum absolute error and .006 average RMS error; and for Ny, .005 maximum absolute and .002 RMS.

Occluding Boundaries

We also wish to know how well the reconstruction process performs when the orientation is not known at all boundary points. In particular, when the surface of interest is occluded by another object, the occluding boundary provides no constraints. In such cases, the orientation at the boundary must be inferred from that of neighboring points, just like at any other interior points of the surface. The 8-way linear operator will correctly handle these situations, since it takes care to avoid interpolating across boundaries. We take advantage of this ability by treating the borders of the orientation array as occluding boundaries, so that we may deal with objects which extend out of the image. For example, spherical surface orientations were correctly recovered from the partially visible boundary shown in Figure 10. The case of the tilted cylinder discussed above is a second example.

Experiments with occluded boundaries raised the question of just how little boundary information suffices to effect recovery. We experimented with a limiting case in which we attempted to reconstruct surface orientation of a sphere from just four initial boundary values at the corners of the arrays. This corresponds to the image of a large sphere whose boundary circumscribes the square array (see Figure 11). The resulting surface orientations produced from these extremely sparse initial conditions were as accurate as when all the boundary orientations are given, but more iterations were required. For example, fixing the Nx and Ny orientations at the corners of a 17 x 17 square array to the values for a sphere of radius 12, the maximum absolute error of the reconstructed interior orientations after 400 iterations was less than .005.

Qualitative Boundary Conditions

In all the above experiments, boundary conditions were provided by specifying exact orientations at all unoccluded points along extremal boundaries. The values of Nx and Ny at these points were initially inserted in the arrays and were held fixed through all iterations. In a complete visual system it is necessary to derive these values from the shape of extremal boundaries in the image. In principle, this can be done easily, since the surface normal at each point is constrained to be orthogonal to both the tangent to the boundary and to the line of sight. (For orthogonal projection, the normal must thus be parallel to the image plane.) In a spatially quantized image, the accurate determination of tangent is difficult, particularly when the object is not very large compared to the quantization grid.

One way to overcome this problem is to introduce the notion of qualitative, partially-constraining boundary conditions. We can, for example, constrain the surface normals along a quantized extremal boundary to be approximately parallel to the image plane and point outward across the boundary. We then rely on the iterative process to reconstruct exact values for the normals at points on the boundary, treating them just like interior points. To implement this approach, we introduce a step which at each iteration checks the orientation at boundary points. For each boundary element adjacent to the point, we check that the surface normal has a component directed outward across it. If it does not, the value of Nx or Ny is modified appropriately. The value of Nz is also checked to be close to zero, and vector N is normalized to ensure it remains a unit vector. This process was applied to the spherical, cylindrical, and elliptical test cases, and was found to yield orientation values accurate to 10 percent, for both interior and boundary points, after only 100 iterations. The principal limitation on accuracy appears to be the coarse quantization grid being used.

FUTURE PLANS

Experimentation is continuing to determine how well the reconstruction technique performs, both in absolute terms and relative to human perception, for a variety of test surfaces. Simultaneously, we are investigating other interpolation operators that reflect measures of curvature appropriate to different surface types, such as soap films. We are also extending the program to deal with a wider range of reconstruction problems, including, specifically, reconstruction from noisy range values and from partially constrained normals along intersection edges, mentioned in the preceding paragraph. These extensions will require properly integrating surface orientation and range (which may require making the integrability condition of Equation 5 explicit), and smoothing noisy, and possibly inconsistent, data. Ultimately, a general vision system will need the ability to add and delete hypothesized discontinuities so that surfaces and boundaries can be simultaneously refined.

Although the reconstruction process we have described is conceptually parallel, there are inherent limitations on how fast information can propagate across the image. Thus, convergence speed is of practical concern. Using larger operators increases the effective velocity of propagation but can impair precision where small features are involved. What seems to be required is a scheme that combines multiple sizes of operators in a hierarchical organization, where initial estimates provided by the larger operators are refined by the smaller ones. We are studying a number of theoretical questions raised by a hierarchical approach to surface reconstruction, including the effects of operator size on speed and accuracy, and the key question of how information propagates between levels of the hierarchy.

CONCLUSION

Interpolating smooth surfaces from boundary conditions is a ubiquitous problem in early visual processing [1, 2, 7, 11-18]. We describe a solution for an important special case: the interpolation of surfaces that are locally spherical or cylindrical, given initial orientation values and constraints on orientation. Our principal contributions are: the observation that components of the unit normal vary linearly on surfaces of uniform curvature; the development of a number of parallel computational techniques for surface reconstruction exploiting this observation; and the clarification of some of the conditions under which surfaces can be reconstructed from incomplete information.

The ability to handle sparse or partially constrained initial conditions is important in a reconstruction algorithm because often nothing else is obtainable. It is well known, for example, that photometric constraints yield families of normals at most points on a smooth surface, not unique values. Also, since range values, as provided by stereo, motion parallax, and laser range-finders, may be noisy, so may initial orientations obtained by differentiating range. A major remaining source of surface information is contour shape, as invoked in the interpretation of surfaces defined by line drawings. In line drawing interpretation [7, 11-13], the initial conditions are extremely sparse, being undefined except along the lines. Moreover, along those lines corresponding to intersection and occlusion (as opposed to extremal) boundaries, orientations are only constrained to be orthogonal to the three-dimensional line segment; their exact directions are indeterminate.

Reconstruction experiments on spherical and cylindrical test cases have produced essentially exact reconstructions, even when boundary values were extremely sparse or only partially constrained. Results on other test cases seem in reasonable agreement with human perception.

ACKNOWLEDGEMENTS

The work reported in this paper was performed under SRI's research program in computational vision, which is jointly supported by ARPA, NSF, and NASA.

REFERENCES

1. H. G. Barrow and J. M. Tenenbaum, "Recovering Intrinsic Scene Characteristics from Images," in Computer Vision Systems, A. Hanson and E. Riseman, eds., pp. 3-26 (Academic Press, New York, New York, 1978).

2. D. Marr, "Representing Visual Information," in Computer Vision Systems, A. Hanson and E. Riseman, eds. (Academic Press, New York, New York, 1978).

3. L. S. Davis and A. Rosenfeld, "Noise Cleaning by Iterated Local Averaging," IEEE Trans. SMC, Vol. 8, pp. 705-710 (1978).

4. R. Haralick, "A Facet Model for Image Data," Proc. IEEE Conf rence on Pattern Recognition and Image Processing, Chicago, Illinois, pp. 485-497 (August 1979).

5. L. Brand, Vector and Tensor Analysis (John Wiley, 1953).

6. H. G. Barrow and J. M. Tenenbaum, op. cit., p. 19, para. 4.

7. A. Witkin, "The Minimum Curvature Assumption and Perceived Surface Orientation," presentation at Optical Society of America, November 1978.

8. F. J. Almgren, Jr., and J. E. Taylor, "The Geometry of Soap Films and Soap Bubbles," Scientific American, pp. 82-93 (July 1976).

9. D. A. Huffman, "Curvature and Creases: A Primer on Paper," IEEE-TC, Vol. C-25, No. 10, (October 1976).

10. C. Wilcox, M. Dageforde, and G. Jirak, "MAINSAIL Language Manual," Stanford University, Stanford, California (July 1979).

11. D. Marr, "Analysis of Occluding Contour," Proc. Roy. Soc. Lond. B, Vol. 197, pp. 441-475 (1977).

12. K. Stevens, "Surface Perception From Local Analysis of Texture and Contour," Ph.D. Dissertation, Electrical Engineering and Computer Science, Mass. Inst. of Technology, Cambridge, Massachusetts.

13. H. G. Barrow and J. M. Tenenbaum, "Recovery of Three-Dimensional Shape Information from Image Boundaries," (in preparation).

14. B.K.P. Horn, "Obtaining Shape from Shading Information," in The Psychology of Computer Vision, P. H. Winston, ed. (McGraw-Hill, New York, New York, 1975).

15. R. Woodham, "A Cooperative Algorithm for Determining Surface Orientation from a Single View," Proc. Fifth Intl. Joint Conference on Artificial Intelligence, Cambridge, Massachusetts, pp. 635-641 (August 1977).

16. M. Brooks, "Surface-Normals from Closed Paths," Proc. Sixth Intl. Joint Conference on Artificial Intelligence, Tokyo, Japan, pp. 98-101 (August 1979).

17. D. Marr and T. Poggio, "Cooperative Computation of Stereo Disparity," Science, Vol. 194, pp. 283-287 (1977).

18. W. F. Clocksin, "Determining the Orientation of Surfaces from Optical Flow," Proc. AISB Conference on Artificial Intelligence, Hamburg, West Germany, pp. 93-102 (July 1978).

TABLE I - INTERPOLATION RESULTS FOR SPHERICAL TEST CASE

Operator	# Iterations	Max. Abs. Error (Nx, Ny)	Average (RMS) error (Nx, Ny)
Uniformly Weighted Average over 4-connected 3X3 patch	50 100	.0165 .0004	.0075 .0002
Uniformly Weighted Average over 8-connected 3X3 patch	50 100	.0007 .0000006	.0003 .0000003
∇^2 over a 4-connected 3X3 patch	50 100	.006 .00006	.003 .00003
8-way linear interpolation/extrapolation (see Figure 6)	50 100	.004 .00002	.002 .00001
4-way linear interpolation/extrapolation (just parallel to x and y axes)	50 100	.03 .001	.01 .0007
Gradient smoothing over a 4-connected 3X3 patch	50 100 200	.40 .26 .10	.19 .12 .05
Gradient smoothing over an 8-connected 3X3 patch	50 100 200	.13 .03 .001	.05 .01 .0005

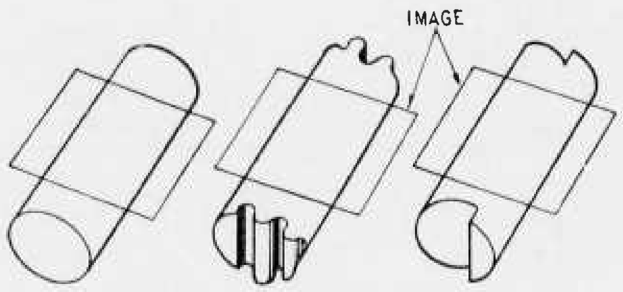


FIGURE 1

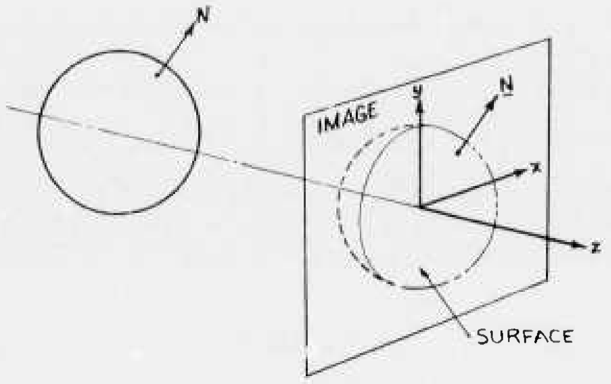


FIGURE 2

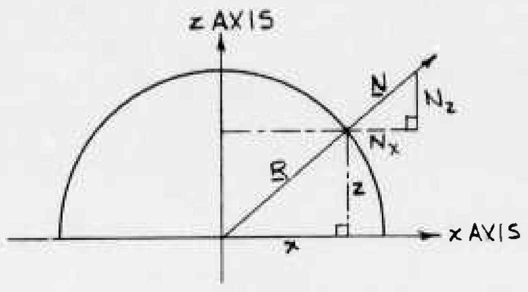


FIGURE 3

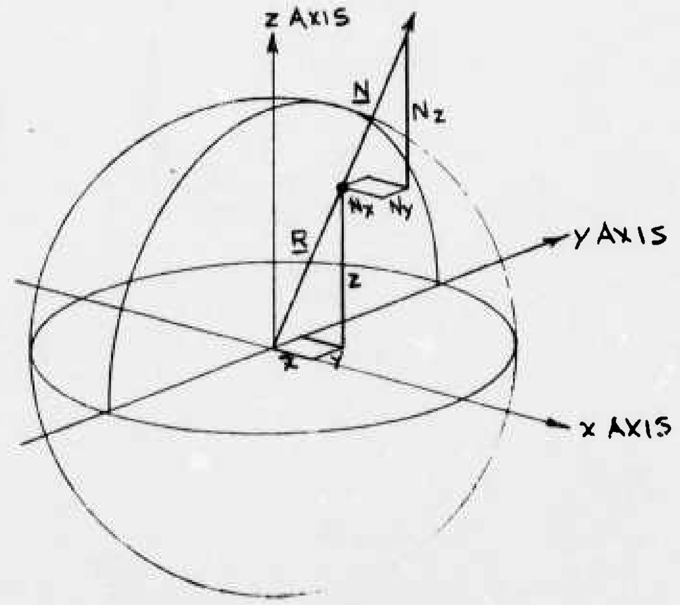


FIGURE 4

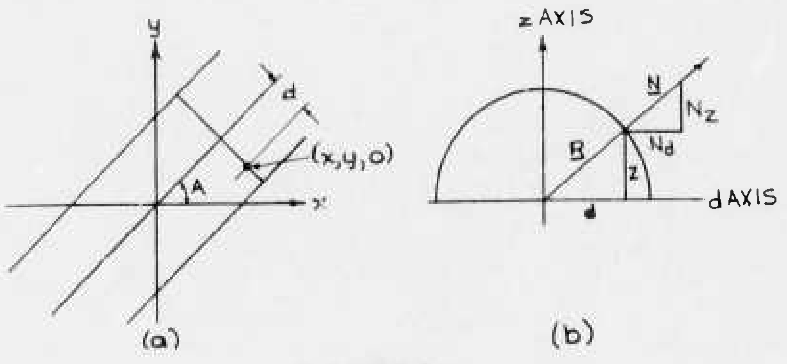


FIGURE 5

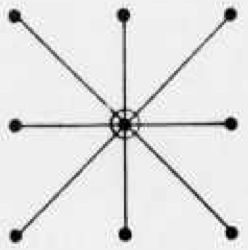


FIGURE 6

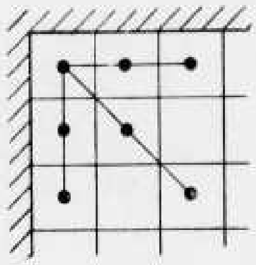


FIGURE 7

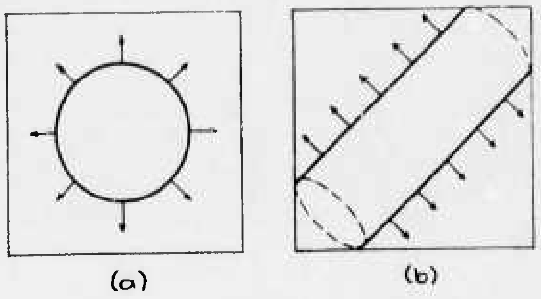


FIGURE 8

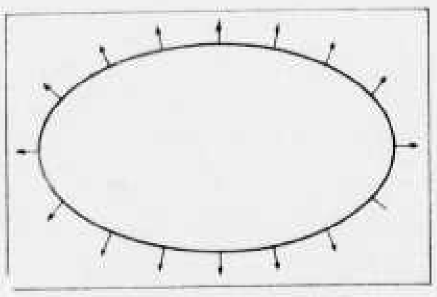


FIGURE 9

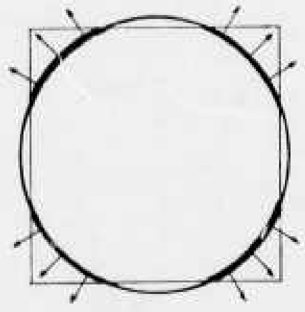


FIGURE 10

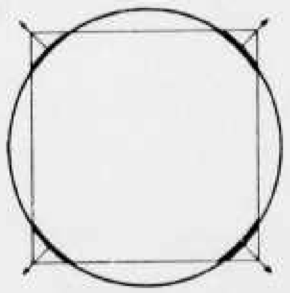


FIGURE 11

DETECTION OF ROADS AND LINEAR STRUCTURES IN AERIAL IMAGERY BY COMPUTER

M. A. Fischler, J. M. Tenenbaum, and H. C. Wolf

SRI International
Menlo Park, California

ABSTRACT

This paper describes a computer-based approach to the problem of detecting and precisely delineating roads, and similar "line-like" structures, appearing in low-resolution aerial imagery. The approach is based on a new paradigm for combining local information from multiple, and possibly incommensurate, sources, including various line and edge detection operators, map knowledge about the likely path of roads through an image, and generic knowledge about roads (e.g., connectivity, curvature, and width constraints). The final interpretation of the scene is achieved by using either a graph search or dynamic programming technique to optimize a global figure of merit. Implementation details and experimental results are included.

INTRODUCTION

A person given the problem of producing an overlay showing the clearly visible roads in an aerial image would normally be expected to accomplish this task with little difficulty, even though he may be completely unfamiliar with the terrain depicted in the image. Our purpose in this paper is to clarify the nature of this task and some of its generalizations. In particular, we wish to specify the requirements and mechanisms for a machine to be capable of near-human performance in finding roads and other semantically meaningful linear structures in aerial images.

Performance Criteria

Our goal is to produce a list of connected points for each segment of road which is tracked in the input image. Each such track is a delineation of the actual road and should have the following properties:

(1) No point on a track should be located outside of the road boundaries when the road is clearly visible.

(2) The track should be smooth where the road is straight or smoothly curving (within the constraints of a digital raster representation).

(3) If parts of the road are occluded, those portions of the continuous track overlaying the occluded segments should be labeled as such.

(4) In areas where the road is partially occluded, the track should follow the actual center of the road (as opposed to the center of the visible portion). If the road is composed of adjacent but separated lanes, then each lane will be considered a separate road for our purposes.

Contextual Settings for Road Tracking

A "road" is a functionally defined entity whose appearance in an image depends largely on its width and how much internal road detail is visible; i.e., appearance depends largely on image resolution (see Figure 1: Road Scenes Depicted At A Spectrum of Resolutions). Additional factors having a major effect on visually locating roads in imagery include the visible extent of the road, its contrast with the adjacent terrain, the presence of nearby linear structures, and any prior knowledge about the actual shape of the road and its location in the image.

We have found that the following contextual settings require significantly different approaches to the road tracking problem:

(1) High vs. low resolution (low resolution is defined as the case in which the road being tracked has an image width of three or fewer pixels).

(2) Clear vs. occluded viewing (clear viewing is defined as a situation in which no more than approximately 30% of the road being tracked is occluded by clouds, intervening objects, etc.).

(3) High vs. low density of linear detail (nominally, this distinction corresponds to urban vs. rural scenes).

In this paper we will mainly be concerned with tracking roads in clear imagery of rural scenes at low resolution. A robust technique for tracking roads in high-resolution imagery was previously reported (Quam [1978]). We note that in the case of high-resolution imagery, once the road has been "acquired" and we are able to track features internal to the road boundaries, the surrounding detail is of minor importance (except as it introduces shadows and occlusions); thus, the distinction between urban and rural scenes is important mainly at low resolution. Where the roads are heavily occluded, road matching rather than road tracking is the appropriate technique; here one needs to have prior knowledge of the geometry of the road networks being searched for. Prior knowledge about the (approximate) location and/or direction of the roads in the imagery is important if a specific road (as opposed to all roads) is to be tracked; some method of indicating which road we are interested in is necessary, and this is typically done by delimiting a search area in the input image. Finally, prior knowledge about terrain type and/or scene elevations can be used to help distinguish low-resolution roads from other linear features by invoking cultural and economic constraints which are known to affect road construction.

LOW-RESOLUTION ROAD TRACKING

At low resolution roads are often indistinguishable from other linear features appearing in the image (including artifacts, such as scratches). Thus, the low-resolution road tracking problem largely reduces to the general problem of line (as opposed to edge) following. Nevertheless, there are still some weak semantics that can be invoked to specifically tailor a system for road tracking, trading some generality for significant increases in performance.

The Basic Paradigm

The basic paradigm we employ is to first evaluate all local evidence for the presence of a road at every location in the search area (a low numeric value indicates a high likelihood that the given image point lies on a road), and then find a single track which, while satisfying imposed constraints (such as continuity), minimizes the sum of the local evaluation scores (costs) associated with every point along the track. While the basic optimization paradigm is not new (e.g., Fischler [1973], Montanari [1971], Martelli [1976], Barrow and Tenenbaum [1975] Rubin [1978]), it is incomplete in that it does not provide mechanisms for reconciling incommensurate sources of information. This capability is crucial in problems such as road tracing in which no single coherent model is adequate for reliable detection. In this paper we introduce new and relatively simple mechanisms for combining local evidence and constraints in the context of an optimization paradigm for detecting linear structures.

Detecting Local Road Presence--Road Operators and Models

At low resolution roads are line-like structures of essentially constant width, which, in general, are locally constant in intensity in the along-track direction and show significant contrast with the adjacent terrain (generally, they are either uniformly lighter or darker). A specific interpretation of this low-resolution road model is embodied in the Duda Road Operator (DRO)* described in Figure 2. In Figure 3 we show some examples of the scores produced by this operator on a variety of road scenes. It is apparent that the DRO does a good job most of the time but has some significant weaknesses; it is sensitive to (a) road orientation (in directions other than the four principal directions explicitly covered by the masks described in Figure 2), (b) raster quantization effects (e.g., where a straight line segment "jogs" in crossing a quantization boundary), (c) sharp changes in road direction, and (d) to certain contrast problems with the adjacent terrain.

At this point one might wonder if a special road operator is really required; why not simply use a generic edge detector (e.g., Sobel [in Duda and Hart, 1973], Roberts [1965], or Heuckel [1971 and 1973])? Even more to the point, we notice that it is possible to interpret the effect of employing an operator on an image as resulting in the suppression of all detail other than that associated with the entity to be detected; therefore, a high-pass filter might act as a perfectly good road operator. Finally, roads will generally be lighter or darker than the immediately adjacent terrain; why not simply use the actual intensity values (contrast-enhanced and possibly inverted, depending on the relative brightness between the road and adjacent terrain)? In Figure 4 we show a comparison of these different techniques applied to the same road scene; in Figure 5 the scores are thresholded to make explicit the locations in the image which are assigned the highest road presence likelihoods by the different techniques.

In the approach we have developed, a key attribute characterizing the utility of a "local" image feature detector (i.e., "operator") is the percentage and coherence of its mistakes when it is almost certain it has found instances of the feature it is designed to detect. Even though the Duda road operator makes mistakes of omission, its performance in not making coherent false-alarm type errors is quite good.

Combining Incommensurate Sources of Knowledge--An Elaboration of the Basic Optimization Paradigm

We will now specify a general approach for combining the results deduced by the application of a set of (road) operators, as well as to the problem of allowing prior knowledge and constraints

----
* Suggested by R. O. Duda of SRI International.

to influence the answer produced by the optimization algorithm.

We partition our inventory of operators into two categories--Type I operators, each of which can be adjusted to make very few coherent errors in detecting instances of the relevant feature when the feature is not present (possibly at the cost of making a large number of omission errors); and Type II operators, each of which can be adjusted to reliably give a quantitative indication of the presences of the feature when it is actually under examination (but these operators might be very unreliable in their assertions when examining something other than the desired feature). Our basic approach is to strongly bias (or even constrain) the desired answer to fit the coherent pattern produced by a superposition of evidence provided by all the Type I operators and to fill in the details locally, using that particular Type II operator which seems to be most certain that it has found the desired feature. (A more comprehensive discussion on methods for combining multisource evidence is given in Fischler and Garvey [in preparation].)

A problem that immediately arises is how to combine the results of several Type I and Type II operators. By considering the output of Type I operators to be valid binary decisions, we have made them commensurate and can logically combine their outputs. In the context of tracking roads (or other linear features), we scan each of our Type I operators over some specified region of interest and create a binary overlay mask containing the logical union of the locations at which one or more of these operators has detected (with high likelihood) the presence of a road. An example of such a mask, called a "perfect road score" (PRS) mask, is compared in Figure 6 with the road image from which it was derived.

The problem of combining the results produced by a set of Type II operators has no acceptable solution when the values they return are not probabilities nor other commensurate quantities. However, Type I and Type II operator scores can be combined, since a positive Type I output can always be set to the maximum value (zero cost) on the likelihood scale of any Type II operator.

Our approach is thus to AND the PRS mask (containing the union of all positive Type I outputs) with every array of scores produced by the Type II operators to produce a set of cost arrays (CA) with zero cost scores at the locations marked in the PRS mask. The optimization algorithm is separately applied to each CA, and the path with the lowest global score is selected as the primary road track through the given region.

In addition to creating a framework for "growing" the road using the Type I operators, we have developed a simple mechanism for introducing constraints and a priori information via the scores obtained from the Type II operators. This is accomplished by numerically transforming the value "x" originally produced by any Type II operator using the function: score = $x\hat{}a+b$ (with control constants a and b). For example, if control constant "a" is held fixed and "b" is increased, the resultant optimal path through the CA would tend to be smoother and straighter (somewhat like pulling the path taut); this effect occurs because, as "b" is increased, the length of the path becomes relatively more important in comparison to the local quality as defined by the individual values "x" returned by the operator. If we are tracking a rocky coastline in an image, we would opt for placing the path through the locations having the best edge scores as opposed to trying to smooth the result; here we would use a zero value for "b". In the case of tracking a road where smoothness is a nominal property, we would select some nonzero value for "b". If we had a priori information that a road we are attempting to track is fairly straight, we could use a high value for "b".

As the value of control constant "a" is increased, there is a very strong inhibition against going through a point having a low likelihood of being on a road. Thus, if we wish to track a road in a region where there may be other strong linear structures nearby, a high value of constant "a" will prevent a jump from one linear object to another; but this can result in wandering (e.g., around shadows, vehicles, etc., in the case of tracking a high-resolution road). Figure 7 shows some examples of tracking a road with different values of the two control constants.

<u>The Low-Resolution Road Tracking Algorithm (LRRT)</u>

The low-resolution road tracking algorithm operates as follows:

(1) A search region is designated in the image. This search region is defined by a binary mask which delimits the search for the road track.

(2) A selected set of Type I operators are scanned over the region designated by the search mask; and the scores produced by each such operator are histogrammed and thresholded at some preset level, based both on the nature of the operator and so that the number of points below this threshold will not exceed the number of road points estimated to be present in the search window (e.g., selecting 5% of the points in the search window might be an upper limit for the Duda road operator). A PRS mask is generated as the union of those locations at which each Type I operator is lower than its associated threshold (scores are treated as costs; a lower score implies a more road-like appearance).

(3) A selected set of Type II operators is scanned over the region designated by the search mask, and the scores produced by

each such operator are either scaled or normalized (e.g., by their histogram ranking) to lie in the nominal range from 1-100; the scores for each Type II operator are stored in a separate array.

(4) Each Type II array is now sequentially modified as indicated:

(a) In those regions of the image where some external source of information indicates that occlusions exist (e.g., due to clouds or to intervening objects), or where there is no significant contrast between the road and the adjacent terrain, thus rendering the road "invisible," a constant is added to the score at each pixel location. This is done in order to reduce the preference for one path over another through areas where the local operators are incapable of returning relevant information about road presence.

(b) The scores at those locations corresponding to points in the PRM are set to zero (actually, they are set to some very small positive value to prevent arbitrary wandering, or even cycling, through regions of zero cost).

(c) Every score "x" in the array is transformed (as discussed earlier) by the formula:

$$x' = x\hat{\ }a+b \quad .$$

This transformation allows us to introduce external information in adjusting the balance between track smoothness (or straightness) and placing the track at its locally most probable location.

(5) Starting and terminating delimiters are designated in the search area: either a pair of lines (e.g., the sides of the search window) or a pair of boxes, through which the road must pass. Each Type II cost array is considered to be a graph with each pixel connected to each of its eight neighbors, and a minimum cost path is found in each such array between the starting and terminating delimiters. Since there is no way to directly compare the relative merits of road tracks computed in two distinct Type II arrays, we employ a heuristic in which the average score per pixel along the track in each Type II array is computed, and its histogram ranking in comparison with all the scores produced by the given Type II operator over the search window is taken as the quality of the track. The track with the highest quality number is chosen as the preferred track.

THE GENERAL PROBLEM OF LOW-RESOLUTION ROAD TRACKING (MULTIPLE ROADS)

We find that it is desirable to deal with the road-(linear feature)-tracking problem in three distinct phases:

(1) The first phase produces a crude delineation of all the roads to be tracked (either producing an approximate track for each road segment or narrowing the search areas containing the different road segments). This delineation can be obtained by making multiple passes through the initial search area of an image with the LRRT described above. After each pass, the detected road track is marked as a forbidden area so as to allow the next most prominent road segment to be detected. If two distinct road tracks have common segments or have the same start and stop delimiters, then the "suboptimal" road tracks produced by the linking algorithm (the algorithm which finds the lowest-cost path through the Type II operator cost array) will generally delineate additional road segments.

With the availability of an external knowledge source, such as a map data base or a sketch map, the desired delineation can be obtained more directly.

(2) The second phase produces a precise track for each road segment of interest by applying the LRRT to the individual crude delineations obtained in the first phase.

(3) The third phase involves smoothing and possibly linking road segments separated by regions of significant occlusion, as well as marking those portions of a road track that were inferred from continuity rather than direct visibility.

IMPLEMENTATION DETAILS AND EXPERIMENTAL RESULTS

While we have addressed the problems associated with each of the above phases for automatically delineating the low-resolution roads and linear structures in an image, most of our current experimental work has been concerned with obtaining a high-performance solution to the problem of precise delineation required in phase two. We have implemented two versions of the LRRT generically described in the preceding section: an INTERLISP/SAIL version for developmental work and a FORTRAN version for more extensive experimentation and evaluation. Both versions run on the SRI PDP-10, while the FORTRAN version was designed to also be compatible with a CDC 6400 system at the U. S. Army Engineer Topographic Lab (ETL) at Ft. Belvoir (the FORTRAN version has a minimum core requirement of 20,000 60-bit words, and will track a road segment 128 pixels long in 15 seconds of CPU time; the corresponding numbers for the INTERLISP version are 90,000 36-bit words of core and 60 seconds of CPU time).

The FORTRAN version of the LRRT makes some additional assumptions about the roads to be tracked: it assumes that they are generally lighter or darker than the surrounding terrain and that they do not "double back" on themselves in the designated search areas. It uses a single Type II operator (based on histogram normalized image intensity) and two Type I operators (the Duda road operator and an image intensity operator, which thresholds image intensity and also checks the size of the above threshold intensity region about a potential road point to determine if the width constraint is satisfied). This program has already been tested on approximately fifty road segments found in aerial images of seven different geographic locations with no failures, where the assumptions are satisfied and the roads are clearly visible (some examples are shown in Figure 8).

CONCLUDING COMMENTS

In this paper we have addressed the problem of precise delineation of the roads and linear features appearing in aerial photographs using an approach based on global optimization of locally evaluated evidence. Since there does not appear to exist a single coherent model suitable for reliable detection of local road presence, it was essential that some means for integrating information from multiple (incommensurate) image operators and knowledge sources be devised--the conventional optimization paradigm does not provide any formal machinery for achieving this task.

Two key points characterize the basis of our approach:

(1) Rather than projecting all image operators on a single linear scale and attempting to use them in the same qualitative manner, we have identified the distinctly different nature and potential use of operators which have strong object detection capabilities as opposed to those which are useful for object analysis once identification and/or location is known. (Depending on the specific context, a particular operator might switch from one role to the other.) We have provided a simple and uniform mechanism for integrating the information provided by the two classes of operators for the specific task of tracking linear structures, and we believe that the same general approach is applicable in a wider range of problem settings.

(2) We have recognized the fact that the score returned by an image operator usually has little absolute meaning, and yet a monotonic transformation of this score can lead to a significantly different final result in tracking linear structures. We have capitalized on this property by introducing a monotonic transform which allows a simple and uniform mechanism for adjusting the scores to reflect a priori information and semantic constraints.

Our plans for future work include the development of more effective techniques for the completely unconstrained delineation required in phase one (defined earlier), for tracking and possibly distinguishing among a variety of different types of linear structures (e.g., roads, rivers, railroads, runways, etc.), and for tracking linear structures in three dimensions using stereo image pairs.

The scientific content of this work lies in discovering effective models for representing and detecting the linear structures of interest and developing paradigms for integrating information from the wide variety of knowledge sources available to the human observer whose performance we are attempting to equal or surpass. Applications of our work in the military area include road monitoring for intelligence purposes, delineation of roads and linear features for automated cartography, and detection of roads and linear features as landmarks for autonomous navigation and weapon guidance.

REFERENCES

1. M. A. Fischler and R. A. Elschlager, "The Representation and Matching of Pictorial Structures," IEEE Trans. on Computers, Vol. C-22, No. 1, pp. 67-92 (January 1973).

2. M. Heuckel, "A Local Visual Operator which Recognizes Edges and Lines," J. ACM, Vol. 20, No. 4 (October 1973).

3. M. Heuckel, "An Operator which Locates Edges in Digitized Pictures," J. Assoc. Computing Mach., Vol. 18, pp. 113-125 (1971).

4. A. Martelli, "An Application of Heuristic Search Methods to Edge and Curve Detection," Comm. ACM, Vol. 19, No. 2 (February 1976).

5. U. Montanari, "On Optimal Detection of Lines in Noisy Pictures," Comm. ACM, Vol. 14, No. 5 (May 1971).

6. L. G. Roberts, "Machine Perception of Three-Dimensional Solids," Optical and Electro-Optical Information Processing, Tippett et al., eds., pp. 159-197 (M.I.T. Press, Cambridge, Massachusetts, 1965).

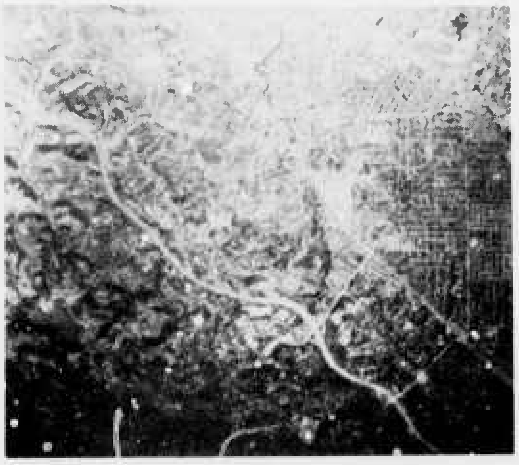
7. L. Quam, "Road Tracking and Anomaly Detection," in Proceedings: Image Understanding Workshop, pp. 51-55 (May 1978).

8. S. Rubin, "The ARCOS Image Understanding System," Ph.D. Thesis, Dept. of Computer Science, Carnegie-Mellon University, Pittsburgh, Pennsylvania (1978).

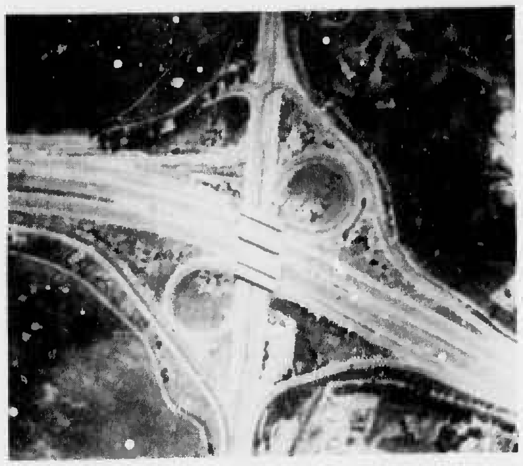
9. H. G. Barrow and J. M. Tenenbaum, "The Representation and Use of Knowledge in Vision," A.I. Technical Note 108, SRI International, Menlo Park, California (July 1975).

10. R. O. Duda and P. E. Hart, Pattern Classification and Scene Analysis (Wiley, 1973).

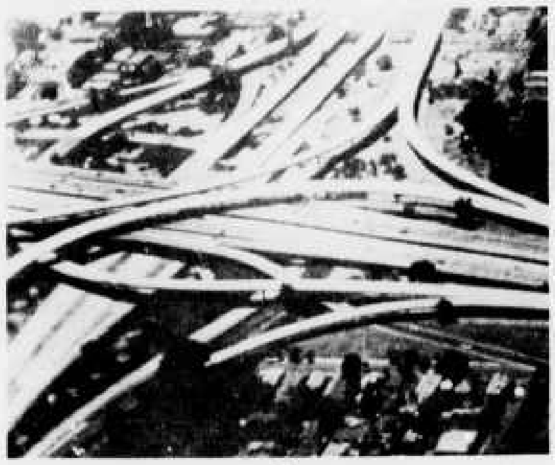
11. M. A. Fischler and T. D. Garvey, "A Computer-Based Approach to Perceptual Reasoning and Multisensor Integration" (in preparation).



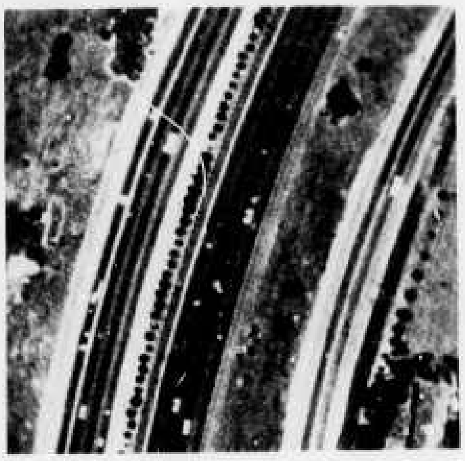
(a)



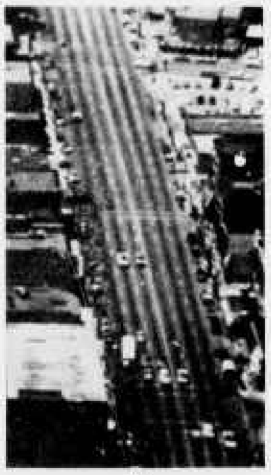
(b)



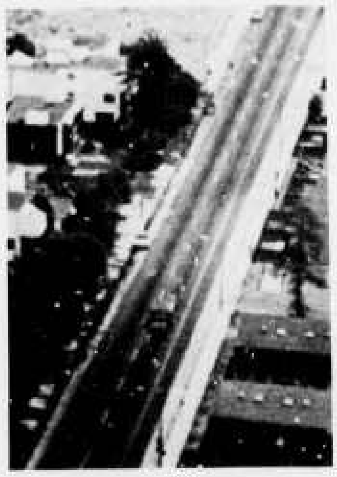
(c)



(d)

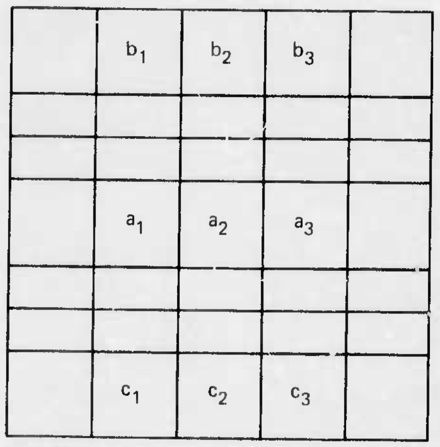


(e)

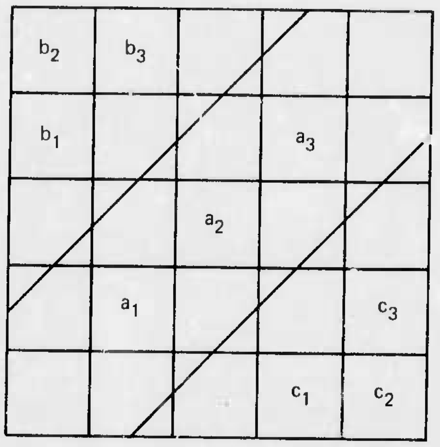


(f)

FIGURE 1 ROAD SCENES DEPICTED AT A SPECTRUM OF RESOLUTIONS



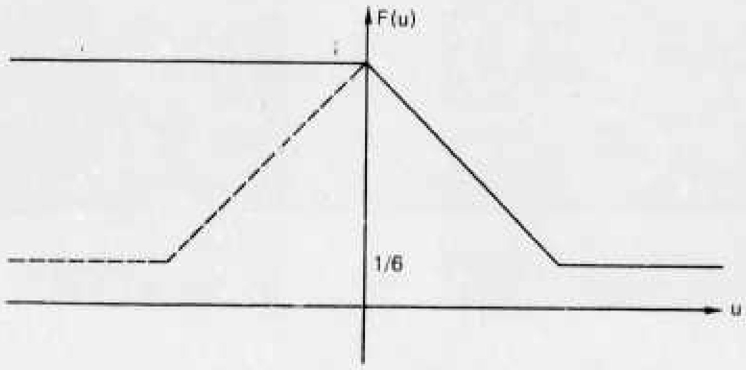
MASK TO DETECT HORIZONTAL ROAD SEGMENTS



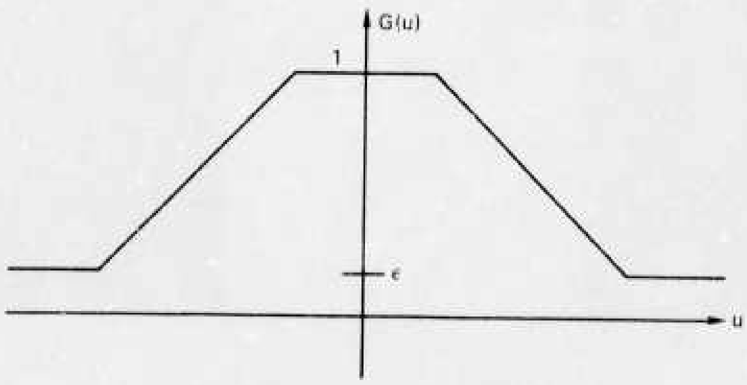
MASK TO DETECT RIGHT DIAGONAL ROAD SEGMENTS

$$\text{SCORE: } 100 - [G(a_1 - a_2) \times G(a_2 - a_3) / \sum_{i=1}^{3} F(a_i - b_i) + F(a_i - c_i)]$$

WINDOWS FOR THE ROAD OPERATORS
Two other masks, similar to these, are used to detect vertical and left diagonal road segments.

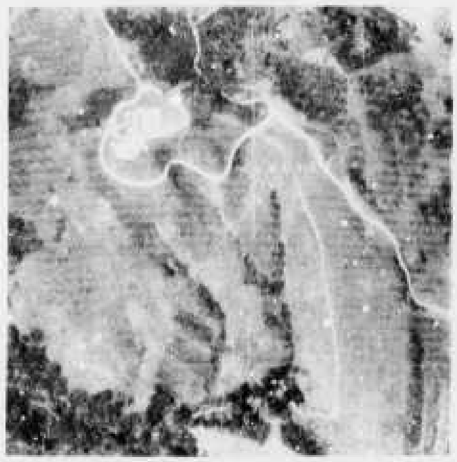


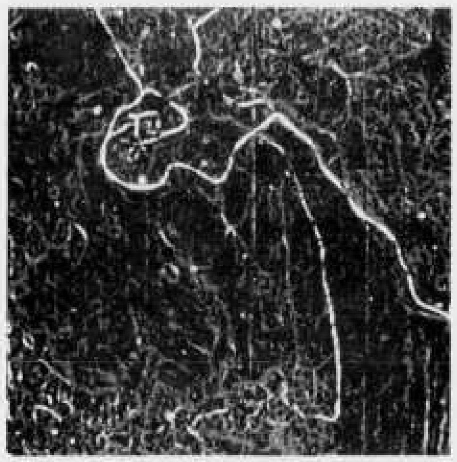
ROAD EDGE SCORING FUNCTION
Function depicted as solid line is used for light roads. F(-u) is used for dark roads. Symmetric form of the function, shown by dashed lines for negative values of u, is used when road to background contrast is unknown.



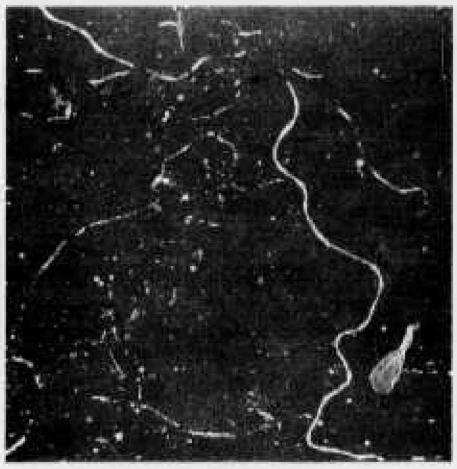
ROAD UNIFORMITY SCORING FUNCTION

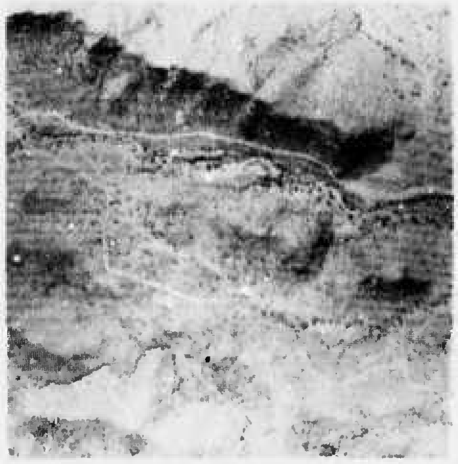
FIGURE 2 DESCRIPTION OF DUDA ROAD OPERATOR


(a)


(b)

(c)


(d)


(e)

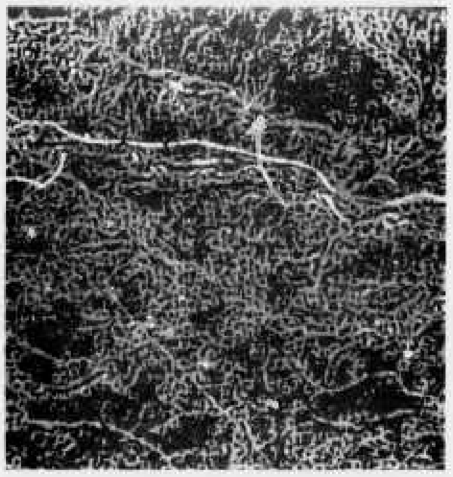
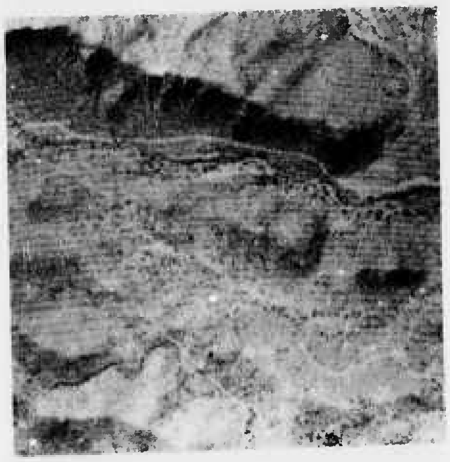
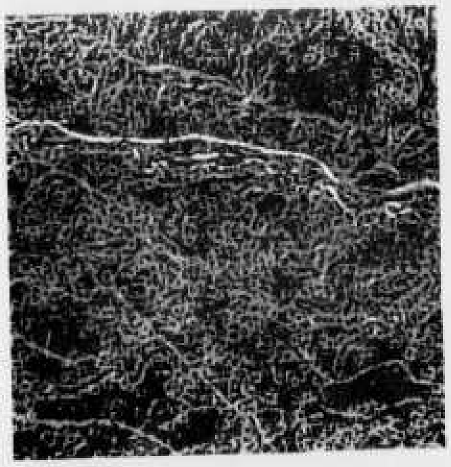

(f)

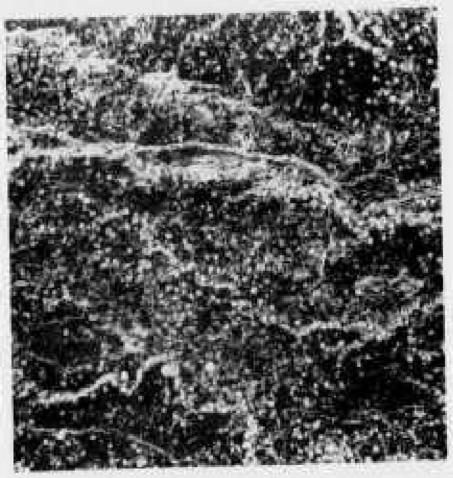
FIGURE 3 DUDA ROAD OPERATOR APPLIED TO A NUMBER OF SCENES



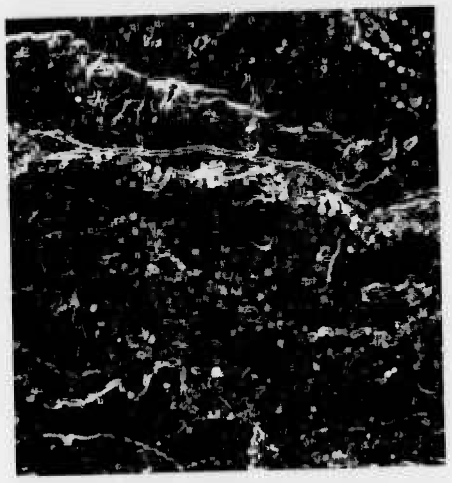
(a) ORIGINAL IMAGE



(b) DUDA ROAD OPERATOR

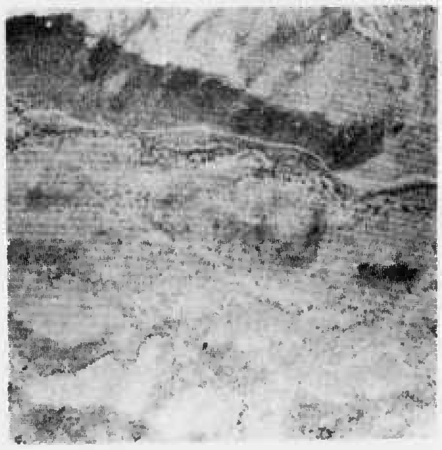


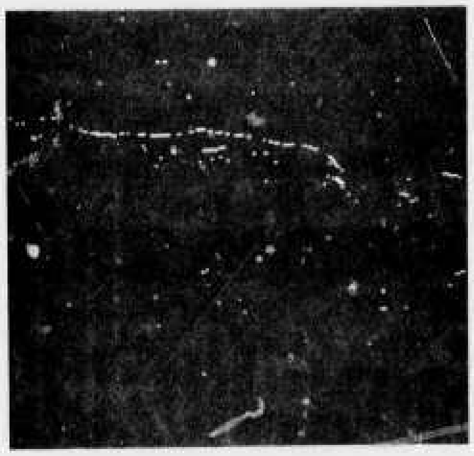
(c) ROBERT'S CROSS GRADIENT

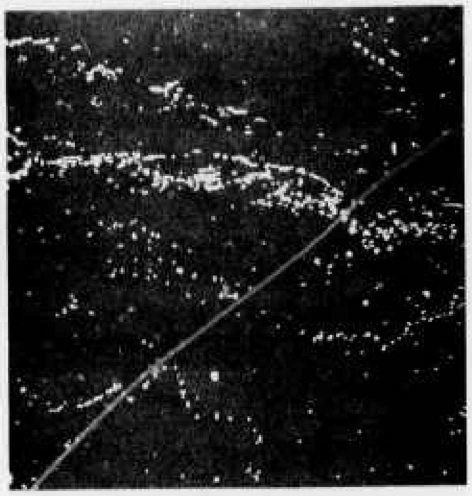


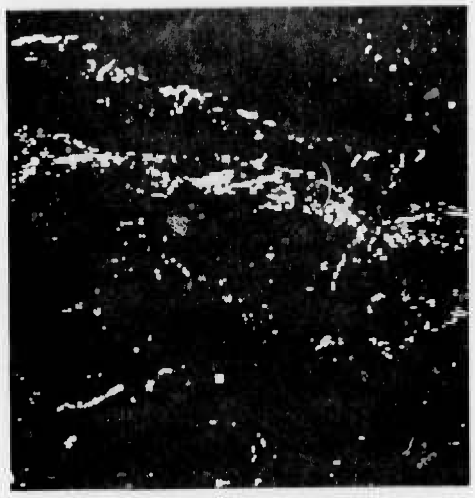
(d) SOBEL-TYPE GRADIENT

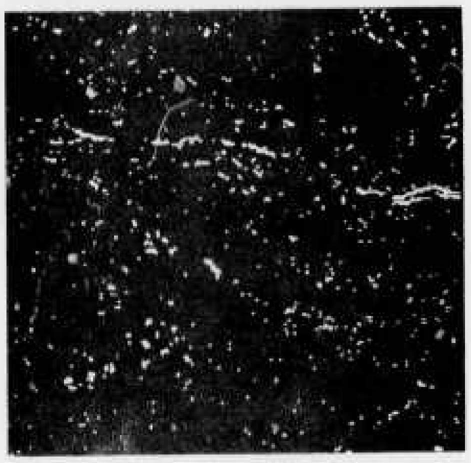
FIGURE 4 DIFFERENT ROAD OPERATORS APPLIED TO THE SAME SCENE


(a) ORIGINAL IMAGE


(b) DUDA ROAD OPERATOR


(c) ROBERT'S CROSS GRADIENT


(d) SOBEL-TYPE GRADIENT


(e) HUECKEL LINE OPERATOR

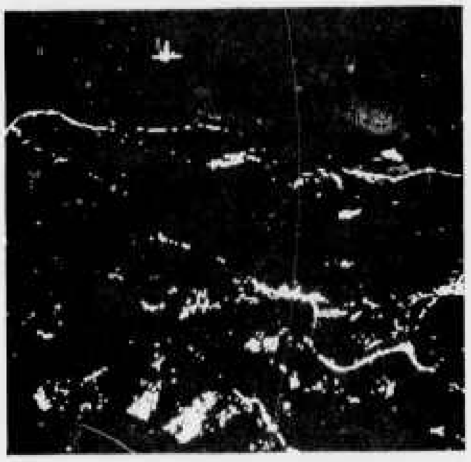
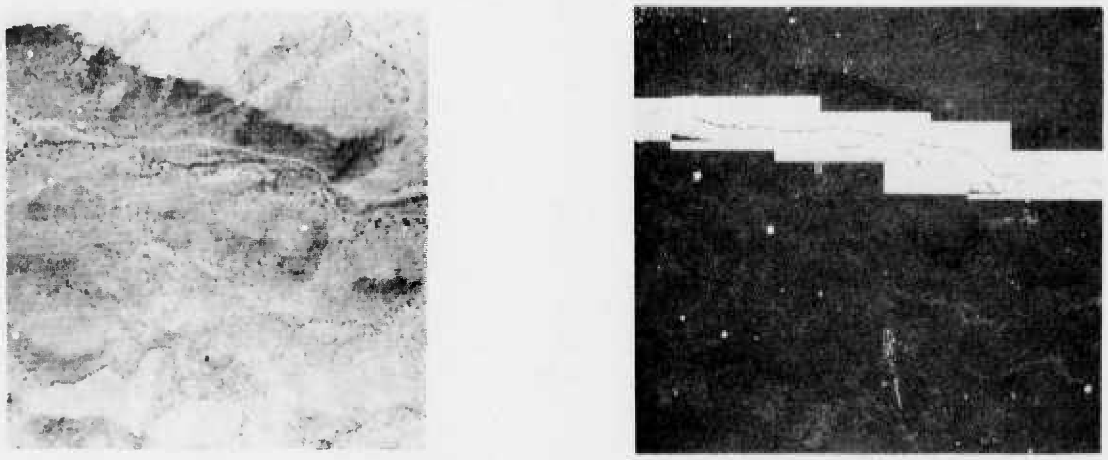
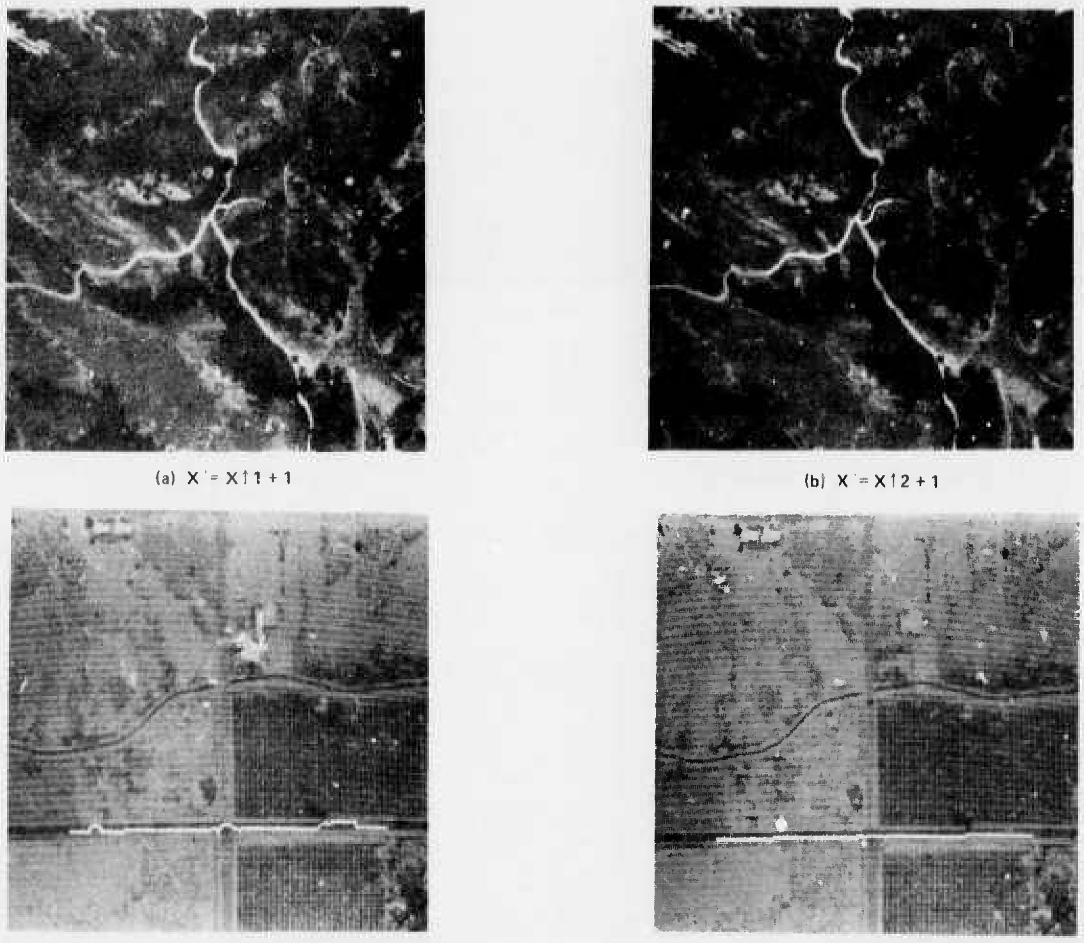

(f) INTENSITY

FIGURE 5 DIFFERENT ROAD OPERATORS APPLIED TO THE SAME SCENE
(Operator scores are thresholded to highlight the locations assigned the best scores.)



(a) (b)

FIGURE 6 A SCENE AND ITS PERFECT ROAD SCORE MASK



(a) X′ = X↑1 + 1

(b) X′ = X↑2 + 1

(c) X′ = X↑2 + 1

(d) X′ = X↑2 + 2000

FIGURE 7 EXAMPLES OF HOW TRANSFORMING TYPE II IMAGE OPERATOR SCORES (X) ALLOW US TO ADJUST THE TRADE-OFF BETWEEN ROAD SMOOTHNESS AND PLACING THE ROAD TRACK AT ITS LOCALLY MOST PROBABLE LOCATION

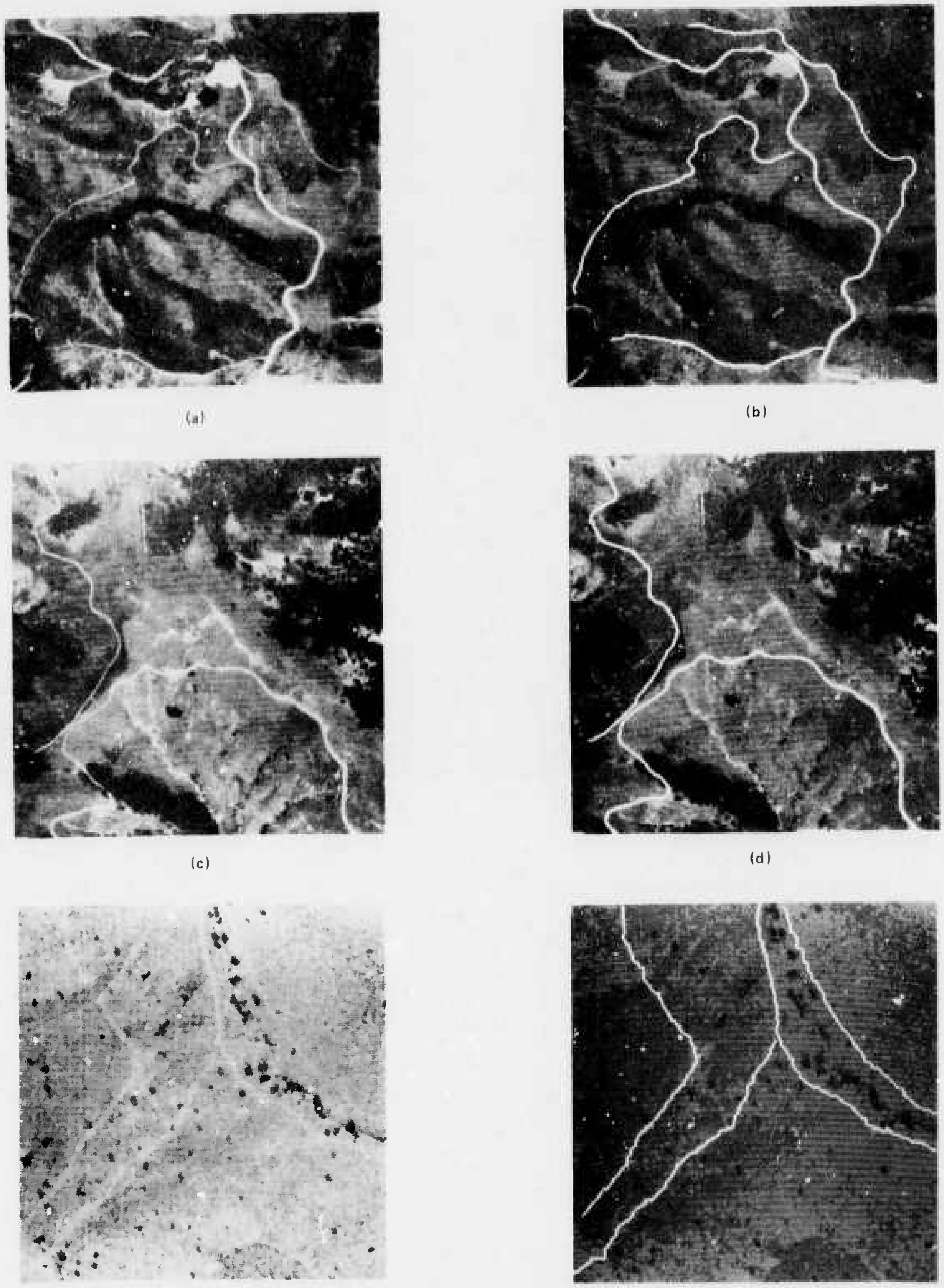
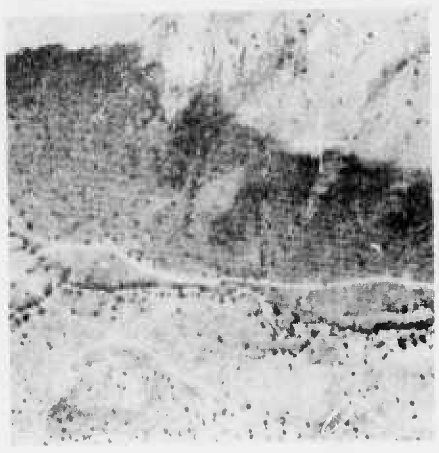
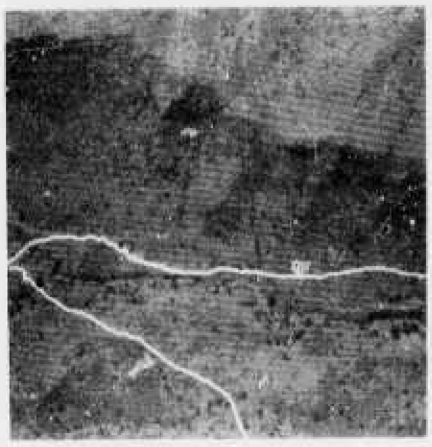


FIGURE 8 EXAMPLES OF ROAD DELINEATION PRODUCED BY THE LOW RESOLUTION ROAD TRACKER

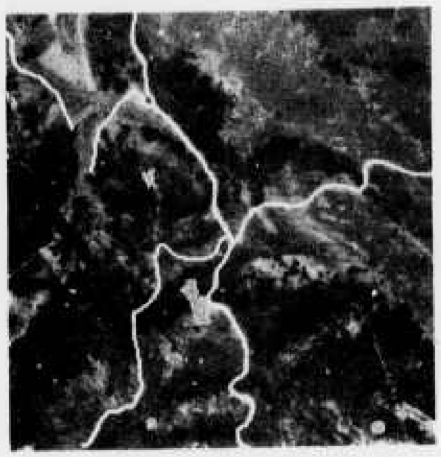


(g)

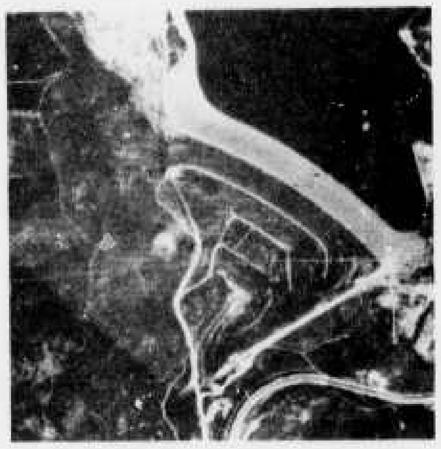


(h)

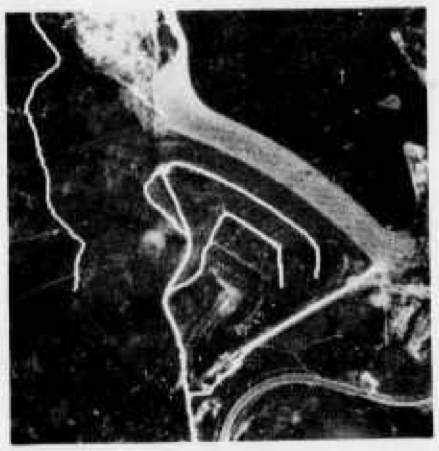
(i)



(j)



(k)



(l)

FIGURE 8 EXAMPLES OF ROAD DELINEATION PRODUCED BY THE LOW RESOLUTION ROAD TRACKER (Concluded)

STEREO-CAMERA CALIBRATION

Donald B. Gennery

Artificial Intelligence Laboratory, Computer Science Department
Stanford University, Stanford, California 94305

ABSTRACT

If the image plane coordinates of several pairs of corresponding points in a stereo pair of images have been measured, it is possible in general to use this information to compute the relative position and orientation of the two cameras, except for a distance scale factor. This paper describes a method of performing this calibration by a generalized least-squares adjustment. First, a general method of performing non-linear adjustments of this type by iterating on a linearization of the problem is reviewed. Then the specific mathematics needed for this problem, using analytical partial derivatives for the linearization, are derived.

I. INTRODUCTION

Suppose we have two camera views of the same three-dimensional scene and wish to extract depth information. Suppose further that the relative position and orientation of the cameras are unknown. The solution then can be divided into three parts. First, corresponding points can be identified in the two pictures. A correlation technique could be used for this step, as described in [1] or [2]. Second, a generalized least-squares adjustment can be done to solve for the relative position and orientation of the two cameras (except that the absolute distance between the cameras cannot be determined unless some distance information is included). Third, the resulting information can be used to compute the (relative) distances to the various points in the image. This paper is concerned primarily with the second of these three steps.

Let Camera 1 denote the camera which will be used as reference. We then wish to compute the direction from Camera 1 to Camera 2 and the orientation of Camera 2 relative to Camera 1. It may also be desirable to compute the focal lengths of the two cameras as part of the adjustment.

Consider any point in the three dimensional scene. Let the coordinates of the image of this point in the Camera 1 film plane be x_1,y_1 and the coordinates of its image in the Camera 2 film plane be x_2,y_2. Image point x_1,y_1 corresponds to a ray in space, which, when projected into the Camera 2 film plane, becomes a line segment. The distance from this line segment to the image point x_2,y_2 is the magnitude of the error in the matching of this point. This error is a function of the angles which define the relative position and orientation of the two cameras (and also the focal lengths). What we desire to do is to perform a camera calibration by adjusting these angles (and perhaps focal lengths) to minimize the weighted sum of the squares (and cross products, if the errors are correlated) of these errors for all of the points that are used.

In Section II a general method of solving nonlinear generalized least-squares problems such as the above by using partial derivatives will be described. In Section III the specific mathematics needed for the above camera calibration problem will be derived.

The method described herein has been implemented as a procedure in CAMRAS[1,DBG] and a driver program CAMRAD[1,DBG] on the PDP-10 at the Stanford Artificial Intelligence Laboratory. This procedure contains several additional features, including variance adjustment, automatic editing, and convergence acceleration, which are beyond the scope of this paper. The original version of the program, without these features or weighting, was written in 1973.

II. GENERAL METHOD

In this section a method of performing nonlinear generalized least-squares adjustments will be described. This method uses partial derivatives to linearize the problem and iterates to achieve the exact solution.

We will use capital letters to denote matrices. Vectors will be represented by column matrices. A particular element of a matrix will be represented by the corresponding lower-case letter with appropriate integer subscripts. The transpose of a matrix A will be denoted by A^T, and the inverse of A will be denoted by A^{-1}.

Suppose we have a set of m unknown parameters for which values are desired, denoted by the vector G (m by 1 matrix). (In our problem, these would be the quantities defining the camera calibration.) Suppose further that there are a set of n quantities ($n \geq m$) denoted by the vector F, which can be measured with some error and which are functions of G. Let U denote the measured value of F (containing some error). (In our problem, the elements of U would be related to the film plane measurements in a way that will by explained in the next section.) Let V be the vector of the n residual errors in the fit to the observations using a particular set of values for the parameters. That is,

$$U = F(G) + V \qquad (2\text{-}1)$$

with the functional dependence on G explicitly indicated. The problem is to use U to compute G such that V is minimized in some sense.

For the criterion of optimization we will minimize the quadratic form

$$q = V^T W V \qquad (2\text{-}2)$$

where W denotes an n by n weight matrix. W should be the inverse of the covariance matrix of the errors in the observations. This will result in the maximum likelihood (in the F space) solution if the errors have the Gaussian (normal) distribution. Note that if W is a diagonal matrix (indicating no correlation between errors in different observations) the quadratic form reduces to a weighted sum of the squares of the elements of V. Thus the problem as stated here can be said to be a generalized least-squares adjustment.

Solving (2-1) for V and substituting in (2-2) produces

$$q = [U - F(G)]^T W [U - F(G)] \qquad (2\text{-}3)$$

The problem then is to find G such that q is minimum.

The difficulty in obtaining a solution to the above problem lies in the fact that F in (2-1) is a nonlinear function, and thus in general there is no closed form solution. One way of solving the problem is to use some type of general numerical minimization technique, in which on various iterations new values of G are tried, q is recomputed each time, and q is driven to a minimum. However, such methods tend to converge rather slowly. Also, numerical problems may occur if q has a very broad minimum, for round-off errors may give rise to spurious local minima. Instead of such an approach, to find the minimum of q, we will differentiate (2-3) with respect to G, set the result to zero, and solve for G iteratively.

In order to follow the steps of this process, we rewrite (2-3) in terms of the elements of the matrices, as follows:

$$q = \sum_{i,j} [u_i - f_i(G)] w_{ij} [u_j - f_j(G)] \qquad (2\text{-}4)$$

Differentiating this produces

$$\frac{\partial q}{\partial g_k} = -2 \sum_{i,j} \frac{\partial f_i}{\partial g_k} w_{ij} [u_j - f_j(G)] \qquad (2\text{-}5)$$

Since (2-5) is a nonlinear equation, to sol e it for G when $\partial q/\partial g_k$ is set to zero, we will use Newton's method. To do this, the partial derivatives of $\partial q/\partial g_k$ are needed. These are

$$\frac{\partial^2 q}{\partial g_k \partial g_l} = 2 \sum_{i,j} \frac{\partial f_i}{\partial g_k} w_{ij} \frac{\partial f_j}{\partial g_l} - 2 \sum_{i,j} \frac{\partial^2 f_i}{\partial g_k \partial g_l} w_{ij} [u_j - f_j] \qquad (2\text{-}6)$$

The corrections d_i needed to g_i are related to the above by

$$\sum_j \frac{\partial^2 q}{\partial g_i \partial g_j} d_j = - \frac{\partial q}{\partial g_i} \qquad (2\text{-}7)$$

(These corrections would be exactly correct if F were linear.) We can now revert to matrix notation, by defining the n by m matrix P to be composed of the partial derivatives of the function F, such that

$$p_{ij} = \frac{\partial f_i}{\partial g_j} \qquad (2\text{-}8)$$

and the n by m by m matrix R to be composed of the second derivatives of F, such that

$$r_{ijk} = \frac{\partial^2 f_i}{\partial g_j \partial g_k} \qquad (2\text{-}9)$$

Substituting (2-5) and (2-6) into (2-7), using these definitions, and dividing through by 2 produces

$$[P^T W P - R^T W (U - F)] D = P^T W (U - F) \qquad (2\text{-}10)$$

where F, P, and R are all implicit functions of G. (An approximate value of G used to obtain F, P, and R in (2-10) defines the correction D needed to obtain a more accurate value.) Notice that R is a strange creature, a three-dimensional matrix. These are not usually defined in matrix algebra, but the usual definitions can be generalized to handle them. In particular, a product of the form $A = R^T W B$, where A, R, W, and B have respectively two, three, two, and one dimensions, is given by $a_{kl} = \sum r_{ilk} w_{ij} b_j$, where the summation is over all values of i and j. (Of the five possible ways of rearranging the three indeces, the transpose of a three-dimensional matrix is defined here as reversing the order of the three indeces.)

The solution for D can expressed in terms of the matrix inverse as follows:

$$D = [P^T W P - R^T W (U - F)]^{-1} P^T W (U - F) \qquad (2\text{-}11)$$

or equivalently

$$D = [I - (P^T W P)^{-1} R^T W (U - F)]^{-1} (P^T W P)^{-1} P^T W (U - F) \qquad (2\text{-}12)$$

where I denctes the identity matrix (in this case m by m). D as obtained above using an approximate value of G would be added to this value of G to obtain a more accurate value, and this process would repeat until it converged.

The worst part of the above solution is the necessity to compute the partial derivatives. Often they are difficult to derive analytically and difficult to compute accurately numerically. In either case they are time-consuming to compute. These difficulties are usually much worse for the second derivatives R than for the first derivatives P. Furthermore, there are nm^2 second derivatives to compute and only nm first derivatives. Therefore, it is highly desirable to be able to omit the second derivatives from the computation. We will now consider the effect of neglecting them.

With a reasonable first approximation, and especially on later iterations, the discrepancies U-F are small. Also, if the function F is reasonably smooth, the second derivatives R are small. Of course, what is considered small is relative. In this case smallness depends on the magnitude of the first derivatives P. If U-F and R are small enough so that the relative change in P is small when G changes enough to cause F to vary by amounts on the order of U-F, then the nonlinearities are not having much effect, and the elements of $R^T W(U-F)$ are small compared to the elements of $P^T W P$. Thus a good approximation in such cases can be obtained by setting R to zero in (2-11) or (2-12), which produces

$$D \approx (P^T W P)^{-1} P^T W (U - F) \qquad (2\text{-}13)$$

The use of this approximation is known as the Gauss method, because Gauss originally used it on ordinary least-squares problems.

The approximate (Gauss) corrections given by (2-13) are just the accurate (Newton) corrections given by (2-11) or (2-12) premultiplied by $I - (P^T W P)^{-1} R^T W (U-F)$. The accurate corrections given by (2-11) or (2-12) attempt to nullify an error in G which Newton's method has estimated to be -D, since -D+D = 0. But, if the Gauss method is used instead, we have in effect $-D + (I - A) D = -A D$, so that the vector of errors in G on each iteration is premultiplied by $A = (P^T W P)^{-1} R^T W (U-F)$, neglecting the higher order effects neglected in Newton's method. Therefore, using the approximation (2-13) cannot effect the final solution, unless it destroys the convergence. The matrix $(P^T W P)^{-1} R^T W (U-F)$ will tend to become constant as the solution convergences, as the discrepancies U-F converge to the final value of the residuals V. Thus the Gauss method changes the quadratic convergence of Newton's method to linear convergence, if convergence is achieved. If all of the eigenvalues of $(P^TWP)^{-1}R^TW(U-F)$ have an absolute value less than one, convergence will be preserved, and the smaller the eigenvalues are, the faster convergence will be. (After several iterations, the error will tend to decrease by a factor equal to the absolute value of the largest eigenvalue.) From the arguments in the previous paragraph, the eigenvalues should be small, except when the initial approximation is very wrong (causing U-F to be large) or when F is very nonlinear (causing R to be large). Thus, except in these cases, the solution should converge rapidly. (A way of converting the linear convergence of the Gauss method into quadratic convergence without computing R will be discussed in a later section.) Some of these matters are discussed further in [3].

The solution using (2-13) is usually obtained by a different approach (as in [4]). This approach approximates (2-1) by a linearization based on the partial derivatives of F, solves the resulting linear problem, and iterates this process to obtain the solution to the nonlinear problem. Thus let G_o denote an approximation to G. Then equation (2-1) can be approximated as follows:

$$U = F(G_o) + P(G_o)(G - G_o) + V \qquad (2\text{-}14)$$

where P is defined by (2-8) and its functional dependence on G has been explicitly indicated. We now define

$$\begin{aligned} E &= U - F(G_o) \\ D &= G - G_o \end{aligned} \qquad (2\text{-}15$$

Then (2-14) can be rewritten as

$$E = PD + V \qquad (2\text{-}16)$$

Thus we have replaced the nonlinear equation (2-1) by the linear equation (2-16), in which E represents the discrepancy between the observations and their computed values using the current approximations of the parameters, and D represents the corrections needed to the parameters. Therefore, we now wish to solve for D in (2-16) so as to minimize q in (2-2). This is a standard problem in linear statistical models. (See, for example, [5].) The solution for D is

$$D = (P^TWP)^{-1}P^TWE \qquad (2\text{-}17)$$

which is the same as (2-13).

The covariance matrix S_G of the errors in the converged values of the parameters G can be obtained from the covariance matrix S_U of the errors in the observations U by the usual linear approximation of premultiplying by the matrix of partial derivatives of the transformation and postmultiplying by the transpose of this matrix. In this case the transformation from U to G in the neighborhood of the converged values is given by approximately (2-13) or more accurately by (2-12). (Regardless of which method was used to produce the converged values of G, the answer is the same. Thus the use of (2-12) will produce a more accurate error propagation than (2-13), although (2-12) is still only an approximation in this regard if higher-order terms are considered.)

If the accurate transformation (2-12) is used, the matrix of partial derivatives will contain terms produced when (2-12) is differentiated relative to both occurrences of U in (2-12). However, when the derivatives are evaluated at the converged values, the effect of the first term drops out, since $P^TW(U-F)$ is then zero (because D is then zero). Thus we have

$$\begin{aligned} S_G = {} & [t - (P^TWP)^{-1}R^TW(U - F)]^{-1}(P^TWP)^{-1}P^TWS_U \\ & P(P^TWP)^{-1}[I - (P^TWP)^{-1}R^TW(U - F)]^{-1\,T} \end{aligned} \qquad (2\text{-}18)$$

If $W = S_U^{-1}$, as it should for the optimum solution, this reduces to

$$\begin{aligned} S_G = {} & [t - (P^TWP)^{-1}R^TW(U - F)]^{-1}(P^TWP)^{-1} \\ & [I - (P^TWP)^{-1}R^TW(U - F)]^{-1\,T} \end{aligned} \qquad (2\text{-}19)$$

Using the approximation of neglecting the second derivatives, as in (2-13), reduces this to

$$S_G = (P^TWP)^{-1} \qquad (2\text{-}20)$$

(Remember that (2-19) and (2-20) are correct only if W is the inverse of the covariance matrix of the observation errors.)

Note that even though (2-19) was derived using the linear approximation for covariance propagation, it contains the second derivatives of F. An even more accurate result could be obtained by considering second-order effects in the propagation, although this would require knowledge of moments of the error distribution of higher order than the second. This result would contain squares and cross products of the second derivatives, whereas they occur to the first power in (2-19). Therefore, if the second derivatives are small, (2-20) and (2-19) can be considered the first two members of an infinite sequence of better approximations, accurate to higher powers of the second derivatives. In most cases (2-20) is quite adequate, since the error estimates usually are not known very accurately anyway.

It often is desired to know the covariance matrix of the residuals. (It is useful to compare the magnitude of the residuals to the square roots of the diagonal elements of their covariance matrix, for editing purposes.) For the approximate case, this can be derived by first obtaining the equation for the residuals by solving (2-16) for V and substituting (2-17) for D, to produce

$$V = [t - P(P^TWP)^{-1}P^TW]E \qquad (2\text{-}21)$$

Then, since the covariance matrix of E is the same as that of U, the covariance matrix of V can be obtained by premultiplying S_U by the coefficient of E (in brackets) in (2-21) and postmultiplying it by the transpose of this coefficient. If $W = S_U^{-1}$, the resulting expression simplifies to

$$S_V = S_U - P(P^TWP)^{-1}P^T \qquad (2\text{-}22)$$

Note that by using (2-20) the second term in this equation is seen to be the covariance matrix of the adjusted parameters propagated into the observations; thus it is the covariance matrix of the adjusted observations. Therefore, (2-22) says that the covariance matrix of the residuals is equal to the covariance matrix of the observations minus the covariance matrix of the adjusted observations. This is seemingly appropriate, because the residuals are the observations minus the adjusted observations. However, this should be considered a coincidence, because the covariance matrix of the difference of two vectors is the sum of their covariance matrices, not the difference, if the vectors are uncorrelated with each other. Here, the particular way in which the observations and the adjusted observations are correlated produces the above result. (Remember that this result holds only in the approximate case and only if the weight matrix is the inverse of the covariance matrix of the observations.)

In many cases W can be partitioned into a diagonal matrix of matrices. Let each of these submatrices on the main diagonal of W be denoted by W_i. In the corresponding manner E and P are partitioned by rows into E_i and P_i. (What we have done is to group the observations into sets so that there is no correlation of errors between members of different sets.) Then (2-17) and (2-20) can be rewritten as

$$\begin{aligned} H &= \sum_i P_i^T W_i P_i \\ C &= \sum_i P_i^T W_i E_i \\ D &= H^{-1} C \\ S_G &= H^{-1} \end{aligned} \qquad (2\text{-}23)$$

Note that, if the errors in all of the observations are uncorrelated, W_i and E_i are 1 by 1 matrices, which can be represented as the scalars w_i and e_i, and P_i is a 1 by m matrix. Furthermore, if all of the w_i are equal, they cancel out of the equation for D, and we have an unweighted solution.

Several other quantities of interest can be derived from the solution. We will present these for the general approximate partitioned case, with $W = S_U^{-1}$. Proofs can be found in references [4] and [5]. The adjusted value of E_i is $P_i D$. The residuals are

$$V_i = E_i - P_i D \qquad (2\text{-}24)$$

The quadratic form is

$$q = \sum_i V_i^T W_i V_i \qquad (2\text{-}25)$$

The expected value of q is n-m. If the scale factor of the covariance matrix of observation errors is unknown, W can be adjusted by the ratio (n-m)/q and S_G by the ratio q/(n-m). Otherwise, q can be used as a test on the adjustment: for, if the observation errors have the Gaussian distribution, then q has the chi-square distribution with n m degrees of freedom. S_G reprents the covariance matrix of errors in the adjusted parameters. The square roots of the diagonal elements of S_G are the standard deviations of the adjusted parameters. The correlation matrix of the parameters can be obtained from S_G by dividing the i,j element by the product of the standard deviations of the ith and jth parameters, for all i and j. Other results which follow directly from the results for the unpartitioned case are the covariance matrix of the adjusted observations $P_i S_G P_i^T$ and the covariance matrix of the residuals $S_{U_i} - P_i S_G P_i^T$.

The solution of the nonlinear problem can now be described as follows. An initial approximation is used to compute the discrepancies E_i and the partial derivatives P_i. Then D is computed from (2-23) and is added to the current approximation for G to obtain a better approximation. This process repeats until there is no further appreciable change in G. Then the final values from the last iteration can he used to obtain S_G, V_i, q, and the other derived quantities described above. Of course, for convergence to the absolute minimum of q rather than convergence to some local minimum or divergence, it is necessary that the initial approximation be sufficiently close to the true solution. In most practical problems this is not critical; in fact, often there is only one minimum.

As previously discussed, the above solution for G, when converged, produces the true generalized least-squares adjustment regardless of the nonlinearity. However, the properties that the solution for G is minimum-variance and unbiased are only approximate in the nonlinear case. Also, as previously discussed, S_G as computed above is only approximately the covariance matrix of the errors in the final value of G in the nonlinear case. However, if the amount of nonlinearity over the range of the measurement errors is small, these results will be fairly accurate.

Often it is desired to have observations directly on the parameters. There are several possible reasons for this. There may be some *a priori* information about the parameters that one wants to combine into the solution. Also, it may be desired to give the initial approximations a very small amount of weight in the solution, so that in case one of the parameters would otherwise be indeterminate, it will be constrained sufficiently to prevent the H matrix from being singular and thus to allow a solution for the other parameters to be obtained. Finally, it may be desired to remove a parameter from the adjustment and to constrain it to a fixed value. This can be done by assigning a very large weight to the given value (although it would save computer time to delete this quantity from the parameters in the program instead). In any of these cases the desired effect can be achieved by creating an additional m by m P_i matrix, say P_o, equal to the identity matrix. Corresponding to this there is E_o, equal to the given *a priori* value of G minus the current approximation of G, and an m by m matrix W_o, the desired *a priori* weight matrix. These are included in the summations for H and C just like any other observations.

A few comments should be made about the numerical aspects of performing the computations. The H matrix is always non-negative definite; that is, if it is not singular it is positive definite. The best strategy to use when inverting a positive-definite matrix by an elimination technique is to pivot on the main diagonal. (See [6].) Therefore, a simple matrix inverter without any pivoting can be used to obtain H^{-1}. H is also symmetrical; therefore, some computation time can be saved if an inverter which makes use of this fact is used. However, if n is considerably larger than m, much more time is spent in computing H than in inverting it, so this is hardly worth the trouble. In problems where the solution is nearly indeterminate, H will be nearly singular, and much accuracy can be lost because of numerical roundoff error. In such cases it may be necessary to use double precision in the computations for H, C, D, and S_G according to (2-23), including the inversion of H. (If a good inverter is used, there is usually not much point in having it in double precision unless a double-precision H is available to invert, as explained in [6].) However, high precision is not needed in computing the discrepancies E_i and the partial derivatives P_i, as long as consistent values are used throughout the computations for H and C.

III. CAMERA MODEL

In this section we describe a method for computing the discrepancies and partial derivatives for the camera calibration problem so that the general solution described in the previous section can be performed. In this method the computations are expressed in terms of matrices and vectors as much as possible, so that the partial derivatives are easy to obtain. In the computer program which uses this method, the matrix operations are performed numerically by standard procedures. Therefore, there is no need to expand these equations to scalar form analytically, except in a few cases where considerable computation time can be saved.

First, the notation used here will be described. Each camera has a Cartesian coordinate system with the origin at the lens center, x to the right in the film plane, y up in the film plane, and z outwards along the optical axis. (The film plane is considered to be in front of the lens center at a distance equal to the focal length.) Thus the coordinate system is left-handed. The azimuth and elevation of the Camera 2 origin relative to the

Camera 1 coordinate system are denoted by α_1 and α_2 (positive to the right from the z axis and up), respectively. The pan, tilt, and roll of the Camera 2 coordinate system relative to the Camera 1 coordinate system are denoted by β_1, β_2, and β_3, (positive right, up, and right), respectively. The focal length of Cameras 1 and 2 are denoted by f_1 and f_2, respectively. Two vectors that will be needed later are defined as follows:

$$T = \begin{bmatrix} x_1 \\ y_1 \\ f_1 \end{bmatrix} \qquad I_z = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{3-1}$$

The symbol $\times$ denotes the vector cross product.

As formulated in the previous section, the general solution method required measurments to be made directly on quantities that are functions of the parameters. However, this is not quite the situation that we have. Here the parameters are α_1, α_2, β_1, β_2, and β_3 (and perhaps f_1 and f_2), and the directly observable quantities are x_1, y_1, x_2, and y_2. Brown [4] describes a way of handling such situations within the general formulation. However, this is not necessary for our purposes here. We will merely propagate the error estimates of the actual observations into the quantity that we use as the discrepancy, in order to obtain the correct weights, and will consider the observations to be measurements directly on the discrepancy on any one iteration. In general, error propagation is done by premultiplying the covariance matrix by the matrix of partial derivatives of the transformation and postmultiplying it by the transpose of the matrix of partials. (This amounts to a linear approximation. Since the discrepancy that we will use will be some distance in the Camera 2 film plane, and since we will consider the measurements to be made in this plane, this transformation is linear and thus the propagation is exact in this case.)

Assume that for each point used in the adjustment an arbitrary point x_1,y_1 in the Camera 1 film plane is picked, and then the position of the corresponding image point x_2,y_2 in the Camera 2 film plane is measured. Let the accuracy of x_2 and y_2 be given by the standard deviations σ_x and σ_y and the covariance σ_{xy}. (The covariance matrix of x_2 and y_2 consists of σ_x^2 and σ_y^2 on the main diagonal and σ_{xy} on both sides off the diagonal.)

The discrepancy e consists of a component of the distance from the point x_2,y_2 to the nearest point of the line segment which consists of the projection into the Camera 2 film plane of the ray defined by the Camera 1 lens center and point x_1,y_1 in the Camera 1 film plane. This is a line segment because a point at an infinite distance on this ray projects into a specific point in the Camera 2 film plane (unless the ray is parallel to the film plane). The coordinates of this infinity point (in the Camera 2 film plane) which defines the end of the line segment are denoted by x_0,y_0. If the point x_2,y_2 is beyond x_0,y_0 (in the direction of the line segment), there are two components of the distance between these two points, and thus there are two observations for this point (two components of the vector E). Otherwise, e consist of the perpendicular distance from x_2,y_2 to the line, and there is only one observation for this point.

The first step in deriving the needed mathematics consists of defining the rotation matrices associated with the angles α_1, α_2, β_1, β_2, and β_3. A prime ($'$) denotes the derivative of the specified matrix with respect to the associated angle. Note that A_1 and A_2 are defined with the opposite direction of rotation from B_1 and B_2. This is because the A's will be used to rotate a vector whereas the B's will be used to rotate the coordinate system.

$$A_1 = \begin{bmatrix} \cos\alpha_1 & 0 & \sin\alpha_1 \\ 0 & 1 & 0 \\ -\sin\alpha_1 & 0 & \cos\alpha_1 \end{bmatrix} \quad A_1' = \begin{bmatrix} -\sin\alpha_1 & 0 & \cos\alpha_1 \\ 0 & 0 & 0 \\ -\cos\alpha_1 & 0 & -\sin\alpha_1 \end{bmatrix}$$

$$A_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha_2 & \sin\alpha_2 \\ 0 & -\sin\alpha_2 & \cos\alpha_2 \end{bmatrix} \quad A_2' = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -\sin\alpha_2 & \cos\alpha_2 \\ 0 & -\cos\alpha_2 & -\sin\alpha_2 \end{bmatrix}$$

$$B_1 = \begin{bmatrix} \cos\beta_1 & 0 & -\sin\beta_1 \\ 0 & 1 & 0 \\ \sin\beta_1 & 0 & \cos\beta_1 \end{bmatrix} \quad B_1' = \begin{bmatrix} -\sin\beta_1 & 0 & -\cos\beta_1 \\ 0 & 0 & 0 \\ \cos\beta_1 & 0 & -\sin\beta_1 \end{bmatrix}$$

$$B_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\beta_2 & -\sin\beta_2 \\ 0 & \sin\beta_2 & \cos\beta_2 \end{bmatrix} \quad B_2' = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -\sin\beta_2 & -\cos\beta_2 \\ 0 & \cos\beta_2 & -\sin\beta_2 \end{bmatrix}$$

$$B_3 = \begin{bmatrix} \cos\beta_3 & -\sin\beta_3 & 0 \\ \sin\beta_3 & \cos\beta_3 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad B_3' = \begin{bmatrix} -\sin\beta_3 & -\cos\beta_3 & 0 \\ \cos\beta_3 & -\sin\beta_3 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{3-2}$$

Now we derive the infinity point x_0,y_0. An image point in the Camera 1 film plane has a three-dimensional position in the Camera 1 coordinate system given by the vector $T = [x_1\ y_1\ f_1]^T$. Since we are concerned at the moment about the infinity point we can ignore the translation between the camera coordinate systems and consider only the rotation. To express the vector T in a coordinate system aligned with Camera 2 we must rotate the coordinate system through the pan, tilt, and roll angles. Let the components of the resulting vector be denoted by the temporary variables u, v, and w. Thus

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = B_3 B_2 B_1 T \tag{3-3}$$

The projection of the point given by the above vector into the Camera 2 film plane is given by a vector in the same direction as the above vector but having a z component equal to f_2. Therefore,

$$x_o = \frac{f_2 u}{w} \qquad y_o = \frac{f_2 v}{w} \tag{3-4}$$

The partial derivatives of $[u\ v\ w]^T$ with respect to β_1, β_2, and β_3 can be obtained by replacing in turn B_1, B_2, and B_3 by B_1', B_2', and B_3', respectively, in equation (3-3). If the partial derivatives with respect to f_1 are desired, they can be obtained by replacing T by I_z in (3-3), since $\partial T/\partial f_1 = I_z$. Equations (3-4) then can be differentiated to obtain the partial derivatives of x_o and y_o, as follows:

$$\frac{\partial x_o}{\partial g} = \frac{f_2}{w}\frac{\partial u}{\partial g} - \frac{f_2 u}{w^2}\frac{\partial w}{\partial g}$$
$$\frac{\partial y_o}{\partial g} = \frac{f_2}{w}\frac{\partial v}{\partial g} - \frac{f_2 v}{w^2}\frac{\partial w}{\partial g}$$
$$\frac{\partial x_o}{\partial f_2} = \frac{u}{w} \tag{3-5}$$
$$\frac{\partial y_o}{\partial f_2} = \frac{v}{w}$$

where g denotes β_1, β_2, β_3, or f_1. (The partial derivatives of x_o and y_o with respect to α_1 and α_2 are zero.)

The point x_o,y_o is the end of the desired line segment. To completely determine it we need also the direction cosines of the line segment (in the direction away from x_o,y_o), denoted by c_x and c_y. These can be found by the following reasoning. The desired line is the intersection of the Camera 2 film plane and the plane defined by the Camera 2 lens center and the ray corresponding to the Camera 1 image point x_1,y_1.

Thus we proceed as follows. The ray which corresponds to the image point x_1,y_1 in the Camera 1 film plane is given by the direction of the vector $T = [x_1\ y_1\ f_1]^T$, in Camera 1 coordinates. First we must determine the plane containing this ray and the Camera 2 lens center. The normal to this plane is given by the direction of the vector cross product of T and the vector giving the direction of the Camera 2 lens center from the origin. This latter vector is $A_1A_2I_z$; that is, the unit z vector rotated through the elevation and azimuth angles. Therefore, the normal to the desired plane is $T \times A_1A_2I_z$, in Camera 1 coordinates. To express this normal in Camera 2 coordinates we must rotate the coordinate system by the pan, tilt, and roll angles. The result is $B_3B_2B_1(T \times A_1A_2I_z)$. The normal to the Camera 2 film plane in Camera 2 coordinates is I_z. The vector along the intersection of these two planes is the cross product of the normals to the two planes, namely $I_z \times B_3B_2B_1(T \times A_1A_2I_z)$. This is the desired line which is the projection of the ray into the Camera 2 film plane, expressed in Camera 2 coordinates. Since the vector lies in the Camera 2 film plane, its z component is zero. Thus, if we redefine u and v as quantities proportional to the direction cosines of the desired line, we have

$$\begin{bmatrix} u \\ v \\ 0 \end{bmatrix} = I_z \times B_3B_2B_1(T \times A_1A_2I_z) \tag{3-6}$$

Application of either the right-hand rule or the left-hand rule consistently to the above two cross products will verify that the above vector has the correct polarity, that is, it points away from x_o,y_o along the line segment. The direction cosines c_x and c_y can now be computed simply as follows from the results of (3-6):

$$r = \sqrt{(u^2 + v^2)}$$
$$c_x = \frac{u}{r} \tag{3-7}$$
$$c_y = \frac{v}{r}$$

The partial derivatives of $[u\ v\ 0]^T$ with respect to the α's and β's can be obtained by replacing in turn the appropriate A's and B's in equation (3-6) by the corresponding A′'s and B′'s from (3-2). The partial derivatives with respect to f_1 can be obtained by replacing T in (3-6) by I_z. Then the partial derivatives of c_x and c_y are obtained as follows, where g denotes any of the parameters (α's, β's, or f_1):

$$\frac{\partial c_x}{\partial g} = \frac{v^2 \frac{\partial u}{\partial g} - uv\frac{\partial v}{\partial g}}{r^3} \tag{3-8}$$
$$\frac{\partial c_y}{\partial g} = \frac{u^2 \frac{\partial v}{\partial g} - uv\frac{\partial u}{\partial g}}{r^3}$$

Now the discrepancy, its partial derivatives, and its weight can be computed. The subscript i will be used to denote to which of the image points these belong, although this subscript will not be used on the other quantities associated with each point, in order to avoid confusion with the other subscripts.

If $(x_2 - x_o)c_x + (y_2 - y_o)c_y \geq 0$, then the point x_2,y_2 does not lie beyond the end of the line segment defined by x_o, y_o, c_x, and c_y, and the discrepancy e_i is the perpendicular distance from the point to the line. Therefore,

$$e_i = (y_2 - y_o)c_x - (x_2 - x_o)c_y$$
$$\frac{\partial e_i}{\partial g} = (y_2 - y_o)\frac{\partial c_x}{\partial g} - (x_2 - x_o)\frac{\partial c_y}{\partial g} - c_x\frac{\partial y_o}{\partial g} + c_y\frac{\partial x_o}{\partial g} \tag{3-9}$$
$$\sigma_e^2 = c_x^2\sigma_y^2 - 2c_xc_y\sigma_{xy} + c_y^2\sigma_x^2$$

where g represents any of the parameters (α's, β's, or f's). (The way in which the polarity of e_i is defined does not matter, as long as the polarity of its derivatives is consistent with this.) If the error in this point is uncorrelated with those of all other points, the correct weight for this point is the reciprocal of its variance:

$$w_i = \frac{1}{\sigma_e^2} \tag{3-10}$$

and the subscript i is the same as i in the summations in equation (2-23).

On the other hand, if $(x_2 - x_o)c_x + (y_2 - y_o)c_y < 0$, there are discrepancies (the two components of the vector E_i) which are the two components of the distance from the point x_2,y_2 to the end of the line segment (x_o,y_o). Any two orthogonal components can be used here; for convenience we will use the x and y components. Therefore,

$$E = \begin{bmatrix} x_2 - x_0 \\ y_2 - y_0 \end{bmatrix}$$

$$\frac{\partial E_i}{\partial g} = \begin{bmatrix} -\partial x_0/\partial g \\ -\partial y_0/\partial g \end{bmatrix} \qquad (3\text{-}11)$$

The covariance matrix of E_i is the same as the covariance matrix of x_2 and y_2. If the error in this point is uncorrelated with those of all other points, the correct weight matrix for this point is the inverse of this covariance matrix:

$$W_i = \frac{1}{\sigma_x^2\sigma_y^2 - \sigma_{xy}^2} \begin{bmatrix} \sigma_y^2 & -\sigma_{xy} \\ -\sigma_{xy} & \sigma_x^2 \end{bmatrix} \qquad (3\text{-}12)$$

and the subscript i is the same as i in the summations in equation (2-23).

Finally, the partial derivatives for each point are assembled into a matrix, as required in the equations in Section II, as follows:

$$P_i = -\left[\frac{\partial E_i}{\partial \alpha_1} \; \frac{\partial E_i}{\partial \alpha_2} \; \frac{\partial E_i}{\partial \beta_1} \; \frac{\partial E_i}{\partial \beta_2} \; \frac{\partial E_i}{\partial \beta_3} \; \frac{\partial E_i}{\partial f_1} \; \frac{\partial E_i}{\partial f_2} \right] \qquad (3\text{-}13)$$

where the last two elements would not be present if the focal lengths are not to be adjusted, and where E_i is replaced by the scalar e_i if the point does not appear to be beyond infinity. (Thus P_i has either one two rows according to whether the first or second case above holds.) The minus sign is present because P was defined in Section II as the partial derivatives of F, which appears in the equation for E with a minus sign.

A few words should be said about the implementation of the above computations. The matrix products A_1A_2 and $B_3B_2B_1$ and their derivatives are computed only once on each iteration. The product of A_1A_2 times I_z reduces to just taking the third column of A_1A_2. The cross products are written out in the code for the program; this reduces the cross product of I_z times another vector to just picking two appropriate terms of the vector, with an appropriate sign change.

Both intuition from the nature of the problem and experience with the program indicate that there are no other local minima of the quadratic form near the absolute minimum. Therefore, if the initial approximation is near the correct solution, the solution should converge to it. However, there is another minimum in cases where one camera is roughly in front of or in back of the other ($\alpha_1 \approx \pm\pi/2$). This local minimum occurs when the front-back positions of the cameras are reversed, for then most of the points appear to be beyond infinity. Tests could be put into the program to detect this condition and to change to the other solution, but this has not been done. If the initial approximation is reasonable, there should be no problem with this phenomenon.

As mentioned in Section II, if the iterative solution converges to the absolute minimum, it produces the exact least-squares solution. Other properties, such as the estimates of accuracy of the adjusted parameters, are approximately correct as long as the problem is approximately linear. This is the case as long as no points are near infinity. However, if a point x_2,y_2 is near the infinity point x_0,y_0 (compared to its standard deviation), a large nonlinearity is introduced. This will cause, among other things, the estimates of error in the solution obtained by the equations in Section 2 to be underestimates if the point x_2,y_2 lies beyond the infinity point or overestimates if the point appears to be closer than infinity. Furthermore, this nonlinearity is caused by a discontinuity. Thus using the second derivatives probably will not help. That is, equation (2-19) may be no better than (2-20) in this case.

REFERENCES

[1] M. J. Hannah, "Computer Matching of Areas in Stereo Images", Stanford Artificial Intelligence Laboratory Memo AIM-239, July 1974.

[2] D. B. Gennery, "A Stereo Vision System for an Autonomous Vehicle", Fifth International Joint Conference on Artificial Intelligence, Massachusetts Institute of Technology, August 1977.

[3] Y. Bard, *Nonlinear Parameter Estimation*, Academic Press, 1974.

[4] D. C. Brown, "A Treatment of Analytical Photogrammetry with Emphasis on Ballistic Camera Applications" (Appendix A, "A Treatment of the General Problem of Least Squares and the Associated Error Propagation"), RCA Data Reduction Technical Report No. 39 (AFMTC-TR-57-22), Patrick AFB Florida, August 1957.

[5] F. A. Graybill, *An Introduction to Linear Statistical Models* Volume I, McGraw-Hill Book Company, 1961.

[6] G. E. Forsythe and C. B. Moler, *Computer Solution of Linear Algebraic Systems*, Prentice-Hall, 1967.

SESSION II

SYMBOLIC REPRESENTATION

REPRESENTATIONS FOR IMAGE DATABASES

David M. McKeown, Jr.

Department of Computer Science
Carnegie-Mellon University
Pittsburgh, PA 15213

Summary

At Carnegie-Mellon we have had a modest research effort for the last two years in determining appropriate and functional representations for *Integrated Image databases*. The need for image databases for knowledge acquisition, photo interpretation and performance evaluation is clear. Traditional approaches to image databases primarily address the problem of efficient access and display of the *signal component* of the image. Part of the reason for this is that inadequate tools are available for describing and representing the *symbolic* aspects of the imagery. This area of research has also suffered because researchers do not tend to work with a large corpus or variety of imagery and have therefore ignored the problems of structuring large image databases. However, photo interpretation systems will have to process thousands of images, produced by a variety of sensors at a continuum of resolutions. Such systems will rely upon *symbolic* knowledge (as well as *signal* knowledge) of imagery, terrain, and cultural features as a basis for its interpretation, and are likely to produce symbolic descriptions as a result of the interpretation process.

For these reasons we must properly address the *Integrated Image database* problem if we are to make progress in building systems for photo interpretation tasks. In this paper we will attempt to outline the *datatypes* inherent in an integrated image database and show that they are useable for a variety of photo interpretation task scenerios.

1. Integrated Image Databases

We will start by describing what it is we wish to have represented in the database and the types of operations and accesses made to the data. Our discussion will be limited to the external view (conceptual view) of the *datatypes*: methods of access, operations on and between datatypes. We will not discuss internal representation or implementation of the database. There are two major points that we wish to make. First, that an *Integrated Image database* should contain (at least) three types of information: image data, terrain data, and map data. Secondly, that each of these datatypes should be viewed along two representational dimensions: the *signal domain* and the *symbolic description domain*.

1.1 Datatypes

In this section we will define our primitive *datatypes*:

Image data consists of collections of digitized images (from photographs or direct sensor output) of vertical and oblique views of areas of interest, taken over time, under a variety of conditions, sensors and signal resolutions.

Terrain data consists of discrete elevations referenced above or below sea level, registered to a known coordinate system. Depending on the application, it may be appropriate to maintain multiple terrain data computed over a variety of grid sizes.

Map data consists of descriptions of cultural features, their structure and composition, size and location. Map features are typically, but not solely, limited to population centers, landforms, roads, railroads, lakes, rivers, power generation and distribution facilities, major industries, dams and bridges.

While we consider the above to be our primitive representational *datatypes* for an integrated image database, it is certainly possible to add other sources of knowledge for photo interpretation such as sensor characteristics and physics, climatic and meteorological data, range data, and task specific data as *datatypes*.

1.2 Representational Dimensions

We can view each of the datatypes as a hierarchy of abstractions along two representational dimensions: the *signal domain* and the *symbolic description domain*. The signal resolution of an image is associated with the digitization aperature or the physical characteristics or optics of the sensor which produced the image. We can obtain a hierarchy of signal descriptions of an image by varying the digitization aperature, or through successive applications of image operators, such as the median, over the image.

For terrain features, the signal domain hierarchy corresponds to the grid size over which the elevation points were computed. The number of levels in the terrain signal description hierarchy is likely to depend on the smoothness of the terrain, i.e., mountainous areas will require a larger number of levels than relatively flat areas.

The signal description hierarchy for map features is analogous to the range of scales at which an area of interest could be mapped. The scale at which maps are drawn and the classes of cultural features they are likely to depict are determined by the function they are to serve. For example, that of a guide for tourists in Washington D.C. or hiking guide in a National Park. The signal description of a map should generate a multi-dimensional descriptor of cultural features on a discrete basis. For each point in the map the signal description contains an indication of the cultural features present at that location. Some likely cultural dimensions are: industry, transportation, storage, vegetation and ground cover and waterways.

The *symbolic resolution* hierarchy for each of our primitive representational *datatypes* can be viewed as collections of abstractions which describe aggregates of signal datatypes. Since the signal resolution determines only the lower bound at which the symbolic descriptors have meaning, there can be (as in the signal domain) a continuum of symbolic resolutions. The symbolic hierarchy for image descriptions is most often described in terms of objects, regions or features. For example, at a coarse symbolic level, an urban aerial scene might be broken into a city region, river region, and an airport region. At a finer symbolic level, regions composed of major roads, industrial areas, residential areas, and park areas might be described, subsumed by the original city region.

The symbolic description hierarchy for the terrain datatype can be represented by aggregation of discrete elevations into topographic features such as hillside, ravine, ridge, peak, and so on. Symbolic terrain descriptions containing the position or orientation of a ridge line, or large regions of local elevation minima, would have utility in the registration of images to maps. Again, symbolic terrain descriptions should be provided for in the database along a range of resolutions.

Symbolic descriptions for maps should allow for the aggregation of map signal descriptions into cultural features. The analogy between *scale* for symbolic map descriptions and *resolution* for symbolic image descriptions is clear. However, since we required that the map signal description produce a multi-dimensional cultural description (ie. dwellings within farmlands, bridges over rivers) it is appropriate that each dimension in the signal descriptor produce a complete and disjoint symbolic map description. The notion of cultural map "overlays" appears to be in harmony with how maps are generated, why they are color coded, and how humans interpret them.

1.3 Datatype Access and Operations

For each of our datatypes: image, terrain, and map, the *Integrated Image database* should provide access to both the *signal* and *symbolic* components of the datatypes.

Signal addressing for image and terrain datatypes is fairly well understood. These datatypes can be stored in a raster organization since many low level image processing algorithms tend to perform a global computation over the entire two dimensional signal. Pyramid organizations can allow efficient access to signal descriptions at a range of resolutions. Block organizations within levels of the pyramid may improve access times to signal elements for certain high level operators. The signal representation of map datatypes is somewhat more difficult since a particular map point may be associated with a variety of cultural features. We propose disjoint "map overlays" with a given overlay representing a specific cultural feature type. These overlays may be efficiently implemented as binary masks.

For most complex photo interpretation tasks the interpretation process will require the ability to retrieve previously processed imagery which covers the area in question, or contains features similar to those being indentified. Thus the ability to find and display *datatypes* both in terms of *content* and *location* should be provided. We have come to call this type of image data operation *symbolic addressing*. For example:

- **image data:** "find all of the images containing bridges over the Potomac river", or "how many images contain the Fort Belvoir area".

- **terrain data:** "show me all ridges whose length is greater than one mile", or "display all plateaus whose elevation is 1000 meters".

- **map data:** "find all forest and farm fields between 50 and 100 acres", or "display all four lane divided highways".

The fundamental *operation* on these datatypes is the *signal to symbol mapping*. This mapping associates a collection of signal descriptions features with a symbolic name. Thus, a collection of image signal description points (pixels) which cover an urban area could be aggregated to form a collection of symbolic entities: power station, harbor area, industrial area, park, etc. The terrain datatype operator maps discrete elevations into symbolic topological features. Similarly the signal to symbol map operator would generate cultural overlays from the map signal.

2. Photo Interpretation Tasks

There exists a variety of tasks which can be addressed as *applications* of an intelligent photographic interpretation system. We intend to use these *datatypes* on some representative task domains: symbolic feature detection, image to map registration, change detection and tracking, map generation and updating, land use and cultural analysis.

3. Examples

We plan to produce for the IUS workshop meeting, Nov. 1979, some examples of a typical *Integrated database* entry for an aerial image centered over Occoquan River in Virginia. These examples will illustrate how each of our primitive *datatypes* can be used to describe major features in the image in both the signal and symbolic description domains.

4. Conclusions

In this paper we have focused on two major aspects of representations for image databases. First, we propose that an *integrated image database* contain image data, terrain data and map data as primitive *datatypes*. Secondly, we believe that each of these datatypes should be viewed along two representational dimensions, the *signal domain* and the *symbolic description domain*. The database should have facilities to maintain features at varying resolutions along each of the representational dimensions, and provide mechanisms for *symbolic accessing* and *signal to symbol mapping*.

LEVELS OF REPRESENTATION IN CULTURAL FEATURE EXTRACTION

Azriel Rosenfeld

Computer Vision Laboratory
Computer Science Center
University of Maryland
College Park, MD 20742

ABSTRACT

This paper describes the extraction of cultural features such as roads and buildings from aerial photographs. It describes an approach based on linking edge pixels into edge segments; grouping these into feature segments such as pairs of anti-parallel edges; and finally extracting global features. This involves three levels of representation, based on pixels, edge segments, and feature segments, respectively. Processing techniques that can be used at each of these levels are described, and preliminary results are presented for small portions of an aerial photograph of the Occoquan, VA area. The approach is also applicable to other types of imagery.

1. INTRODUCTION

This paper describes an approach to the extraction of cultural features such as roads and buildings from aerial photographs. The approach involves three stages, at which successively more global knowledge about the features is used to guide the extraction process. Each stage uses an appropriate type of data representation; arrays of pixels, sets of edge segments, and sets of feature segments are used at the successive stages.

In the following sections of the paper, the stages of the feature extraction process, as they are presently conceived, are described. The reasons underlying various design decisions are given; this subject is also discussed in the current status report on this project, elsewhere in these Proceedings. Examples of results obtained at the first two stages are also given. These examples involve small portions of an aerial photograph of the Occoquan, VA area, distributed by DARPA to the participants in the Image Understanding Program.

The approach taken in this paper was motivated by the following considerations:

a) It is necessary to develop methods that can deal with cases where map information, giving the approximate locations of the features to be extracted, is unavailable. The SRI approach to road finding [1] does make use of such information. Evidently, however, there will be many situations where map information is not available or is unreliable, particularly in cases involving features of a transient or short-lived nature, or areas that are not frequently mapped at the desired level of detail.

b) An effort has been made to use methods that can be implemented by parallel processing techniques, particularly at the lower levels. If inherently sequential methods, such as road tracking, are used too extensively, it will be difficult to implement the feature extraction process in real time.

c) In order to reduce computational costs, the approach has been broken up into stages, at which increasingly global and more specialized knowledge about the features to be extracted is used. The first stage involves local operations on pixels, using general information about the local properties (gray level, color, contrast, etc.) that pixels belonging to the features are likely to have. Since at this stage we are examining every pixel, it is important that only simple computations be performed. The principal output of this stage is a set of line segments representing fragments of feature edges, and labelled with various property values computed for these fragments. The second stage groups these edge segments into pieces of features ("feature segments"), based on "semi-local" properties of the features (curvature, parallel-sidedness, etc.); the third stage groups the feature segments into global features, using global information about their shapes and spatial relationships. Thus at each stage, the computations are more complex, but they are applied to a smaller set of data.

d) Since the approach involves several successive stages of segmentation or grouping, if errors are made at an early stage, they may be difficult to correct at later stages. It is important to preserve the correspondences between entities at successive levels--i.e., between edge segments and the pixels that comprise them, and between feature segments and the edges of which they are composed; this will make it easier to locate the sources of any errors. It is also highly desirable to avoid firm decisions at any stage, and to avoid the use of processes that involve thresholds, but rather to make fuzzy or "probabilistic" decisions whenever possible, thus deferring commitments until they are confirmed by corroborating evidence. Note that when firm decisions are made, inputs that differ by arbitrarily small amounts may give rise to drastically different outputs. If such decisions must be made, they should be based on as much information as possible.

2. THE PIXEL LEVEL

Cultural features often contrast with their surrounds, and are usually bounded by sharp, locally straight edges. These characteristics can be used as guidelines in classifying pixels as possibly belonging to such features. On the other hand, information about feature shapes and spatial relationships would normally not be very useful in making decisions about pixels, unless the information is very specific, i.e., template-like. Knowing that houses are rectangular, for example, does not help us in classifying a pixel as being possibly part of a house, since we do not know its position in the house, so that we can say very little about how it should be related to other pixels if it is indeed part of a house.

If the features have characteristic gray levels or colors, we should certainly use these properties in making decisions at the pixel level; but in non-multispectral imagery, it will usually not be possible to characterize features in this way. Moreover, if we do classify the pixels based on their gray levels, we will often obtain large connected components of constant gray level; thus using a very local classification criterion (the pixel's gray level) may give rise to relatively global segments, and this will often be unwarranted.

These considerations have led us to propose the use of an edge-based approach at the pixel level. We first use local operators to estimate the magnitude and direction of the gradient at each point. We then use an iterative process at the pixel level to adjust the magnitudes and directions in the following way:

a) The magnitude is increased in the presence of high magnitudes at neighboring points in the tangential direction, provided their directions are smooth continuations of that direction; and it is decreased in the absence of such neighbors. This strengthens the edge responses at points that lie on straight or smoothly curved edges, and weakens them elsewhere.

b) At the same time, the direction is adjusted to make it agree more closely with these neighboring directions; the amount of adjustment depends on the magnitudes at these neighbors. This tends to smooth out irregularities in the edge responses caused by noise.*

*An iterative scheme could also be used [2] for edge thinning: The magnitude is reduced in the presence of higher magnitudes at neighboring points in the gradient direction, and increased in the presence of lower magnitudes. If this is done iteratively, the magnitudes at the tops of the "ridges" of responses increase, while those at other points decrease, so that the edge responses are thinned.

Thus this process should produce sets of high-magnitude responses that lie on (thin) straight (or smoothly curved) edge segments, and such that the associated directions are locally very consistent. Note that the process involves no thresholds or decisions, and that it is readily implementable in parallel.

Figure 1 illustrates the results of applying such processes to the edge responses in a small portion of the aerial photograph. The desired enhancement effects are all quite apparent.* The specific algorithms used were described in an earlier technical report [3]. Many variations on these algorithms could have been used, and would have yielded similar results; e.g., see [4]. An edge enhancement relaxation scheme could also have been used.

The approach to feature extraction at the University of Southern California [5] is also edge-based, but it involves a one-pass process of nonmaximum suppression and thresholding, rather than an iterative, quantitative process. The USC approach is thus computationally cheaper, but it is probably more likely to make errors.

3. EDGE SEGMENT CONSTRUCTION

We now want to construct a data representation based on entities more global than pixels; this will allow us to use more global knowledge about cultural features, e.g., simple types of shape information, to classify these entities. Straight or smoothly curved edge segments are obvious choices for these entities, since the pixel-level processes tend to produce sets of edge responses that lie along such segments.

Extracting edge segments inherently involves some sort of threshold criterion, since one must decide whether or not to construct a segment corresponding to a given collection of edge responses. Such decisions should be easier for enhanced responses, but they are still nontrivial, and should be made on the basis of as much information as possible. If we simply threshold the (enhanced) edge magnitudes, we are making the decisions on a pixel by pixel basis, using only the information concerning that pixel, which is undesirable. (Note, however, that when we do this for enhanced responses, the information associated with a pixel also reflects the nature of its neighbors.)

* Since no thinning was done, the magnitude reinforcement process tends to thicken the edges; but this is not considered harmful, since in any case line segments will be fitted to the edges at the next step, and these will be much the same whether or not the edges are thin--in fact, they may be more reliable if the edges are thick.

A somewhat safer idea is to make decisions about pixels in the context of their neighborhoods. For example, one might "accept" a pixel if its own magnitude, and the magnitudes of two of its neighbors in the tangential directions, are sufficiently high. (Note that this idea is very compatible with the enhancement process; it essentially accepts just those pixels that would be strongly enhanced.) At the same time, one can establish links between each accepted pixel and its neighbors; these links can then be used to define connected components of accepted pixels, which then constitute the desired edge segments. Such a linking approach is used by USC [5]. Alternatively, once can use a global straightness criterion in defining the connected components by requiring each pixel's direction to lie close to the average direction of the already accepted pixels [3]; this breaks up smooth curves into segments having relatively low net changes in slope from one end to the other. Figure 2 illustrates the types of edge segments obtained using this criterion.

It would be even more desirable to make decisions about entire groups of linkable edge pixels; but the number of such groups is enormous, and it is utterly impractical to consider all of them. However, suppose that we are only interested in groups of edge pixels that lie on a curve of a given shape, e.g., on a straight line. In this case we can use a Hough transform approach to map collinear sets of edge responses into compact peaks in the Hough space. We must then use a threshold criterion to detect the peaks, but this criterion is now being applied to an entire group of collinear edge pixels, rather than on a pixel hy pixel basis. It should be mentioned that we obtain a cleaner Hough space when we use enhanced edge responses, since the slope estimates are much more consistent than in the raw responses, and this in turn makes the estimates of the distances of lines from the origin much more consistent. Of course, we should not merely use slope and distance (and response magnitude) to define clusters in Hough space; other properties associated with the edge responses, e.g. the gray levels on the two sides of the edge, should also be used if appropriate, to differentiate between responses that (probably) belong to different edges. It may even be desirable to use position along the line as a feature, in order to avoid clustering responses that are far apart in the image and have no responses between them. This more global, Hough-like approach to edge segment construction is currently under investigation.

4. THE EDGE SEGMENT LEVEL

We now have a set of edge segments, with each of which we can associate various properties, including its length, average slope, average strength, etc., as well as properties of the gray levels on the two sides of the segment's constituent edge pixels, e.g. the means and standard deviations of these gray levels. If desired, we can now use this information to search for missing parts of edges in the original image, so as to fill gaps in the edge segments and create longer ones. We can also now group the edge segments into linear feature segments, based on our knowledge about the expected geometrical properties of these segments. In this section we discuss some possible approaches to edge segment grouping. For simplicity, we consider two simple types of grouping, based, respectively, on good continuation and on parallelism.

Straight segments that are collinear, or curved segments that "point toward" one another, can be linked using criteria based on strength, length, distance, and good continuation, as well as similarity of properties [6]. (This assumes, of course, that such linking is consistent with what we know about the features that we are trying to extract.) Linking across large gaps can be done much more reliably at the segment level than at the pixel level, since the information that we have about the segments (slope, property similarity, etc.) is more reliable than the corresponding information about pixels. At the same time, exploration of large gaps at the pixel level would involve an excessive amount of computation per pixel.

This type of linking involves pairwise decisions; as pointed out in Section 3, it would be preferable to make decisions about entire groups of segments as to whether or not they constitute good groupings, rather than making decisions about two segments at a time. In general, it is not practical to consider all possible combinations of segments; but if we restrict ourselves to sets of collinear segments (or more generally, segments that lie on a curve of known shape), it is computationally feasible to evaluate all possible pairs of consecutive segments as possible groupings. Various criteria for evaluating such groupings are under investigation. The problem is analogous in some respects to that of peak detection in waveforms, since clusters of segments can be regarded as peaks in the density of segments along the line. Simple criteria for clustering collinear segments will often yield good results; for example, Figure 3 shows the results of combining pairs of consecutive segments that are separated hy gaps shorter than the sum of their lengths. Note that this process can be iterated.

In addition to segment linking based on collinearity or good continuation, one usually also wants to link pairs of "antiparallel" segments, representing pairs of parallel edges whose dark sides or light sides face one another, since cultural features often have parallel sides. In the USC experiments [5], links are shown only for pairs having no segments between them; but in general, we should be allowed to link two segments even if there are other segments between them, since these other segments may be due to noise, or may represent features internal to the given one (e.g., a penthouse on a building, a divider strip on a highway). Thus in general we must compute link merits for many pairs of segments, and then choose "best" pairs for actual linking. The merit function may depend on the strengths, slopes, lengths, and property value similarity of the segments, as well as on their degree of overlap and on the distance between them, and on any special knowledge that we may have about the properties of the desired features. A number of simple merit functions are currently under investigation. Note that the merit may be asymmetrical; for example, if a short segment and a long segment face one another, the merit of linking the short one to the long one may be much higher than that of linking the long one to the short one. Given the merits for all pairs of segments, we can link all pairs having (mutually) highest merit; once we have done this, the linked segments are no longer candidates for linking, so that some of the remaining pairs may now have mutually highest merit and can now be linked. This process can be repeated until no further linking is possible. Figure 4 shows the results of applying this process using a very simple merit function, namely the fraction by which one segment overlaps the other divided by the distance between them, provided the segments have approximately equal slopes. Several variations of this approach have also been tried, with essentially identical results.

The antiparallel linking schemes just described are all based on pairwise decisions. As before, it would be preferable to evaluate groupings of segments that form antiparallel strips, rather than linking such segments two at a time. This would allow us to combine the collinear and antiparallel linking processes into a single strip clustering process. Here again, a Hough-like approach might be used to detect clusters arising from strips. Such an approach is currently under investigation.

5. THE FEATURE SEGMENT LEVEL

The result of the linking processes at the edge segment level is a set of groups of edge segments--e.g., antiparallel strips--that hopefully constitude significant pieces of features--wings of buildings, uninterruped segments of roads, etc. With each of these feature segments, a set of geometrical and gray level properties is associated; in addition to properties inherited from the constituent edge segments, these should include textural properties of the region defined by the strip and of the regions bordering it on each side. Our final goal is to classify these segments in terms of models for the sets of features that we expect will occur, and at the same time, to link the segments into global features (buildings, roads, etc.). This should also allow us to correct errors on the edge segment (and pixel) levels by predicting missing segments that are required by the models and eliminating segments that contradict the models.

We plan to investigate a relaxation-like (or MSYS-like) scheme for classifying the feature segments. Initially, each individual segment will be probabilistically classified, on the basis of its properties, as being (part of) a road, building, etc. These probabilities will then be adjusted based on their compatibilities with those of nearby or otherwise related segments. One should not expect that a simple algebraic formula can be used to compute the probability adjustments; rather, they will be computed by a probabilistic "decision tree" associated with each segment. This approach should result in a generally consistent classification (which, of course, may still be ambiguous). If inconsistencies remain, they will probably reflect errors in the feature segment extraction process, assuming that the compatibility models are adequate.

6. CONCLUDING REMARKS

This paper has outlined a proposed approach to the extraction of cultural features such as buildings and roads from aerial imagery. The later steps in the approach have not yet been designed in detail, and some of the methods currently implemented at the earlier stages represent compromises made on grounds of expediency. However, further work is being done at all levels, and it is expected that a reasonable approximation to the system outlined above will be implemented by mid-1980. Tests of this system, and of variations on it, will then be conducted on DARPA-supplied imagery.

REFERENCES

1. M. J. Fischler, G.J. Agin, H. G. Barrow, R. C. Bolles, L. H. Quam, J. M. Tenenbaum, and H. C. Wolf, The SRI road expert: an overview. Proceedings, Image Understanding Workshop, November 1978, 13-19.

2. R. B. Eberlein, An iterative gradient edge detection algorithm, Computer Graphics Image Processing 5, 1976, 245-253.

3. S. Peleg and A. Rosenfeld, Straight edge enhancement and mapping, Computer Science TR-694, University of Maryland, College Park, MD, September 1978.

4. G. J. VanderBrug, Experiments in iterative enhancement of linear features, Computer Graphics Image Processing 6, 1977, 25-42.

5. R. Nevatia and K. Babu, Linear feature extraction, Proceedings, Image Understanding Workshop, November 1978, 73-78.

6. G. J. VanderBrug and A. Rosenfeld, Linear feature mapping, IEEE Trans. Systems, Man, Cybernetics 8, 1978, 768-774.

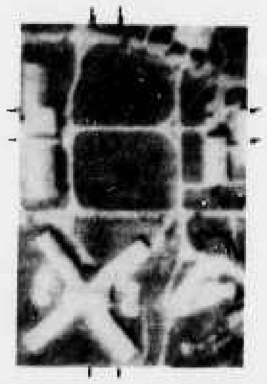


Figure 1. (a) Window of the Occoquan photograph, showing parts of Lorton reformatory

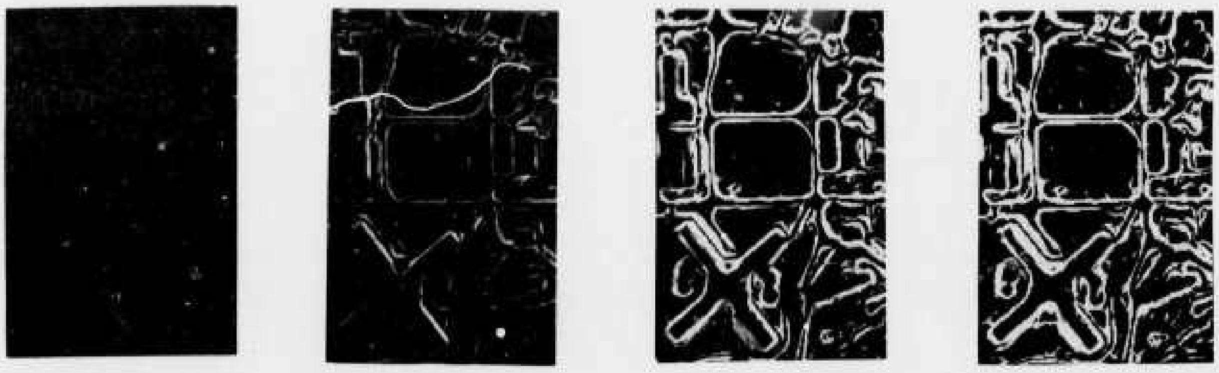


Figure 1. (b) Original Sobel gradient magnitudes and three iterations of the enhancement process

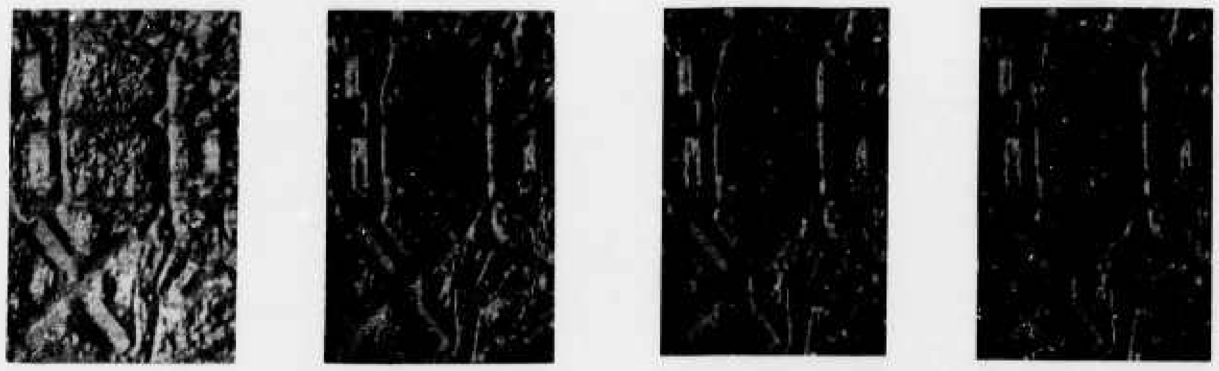


Figure 1. (c) Gradient directions, displayed as gray levels ranging from black to white as the direction (from dark to light) varies from 0° to ±180°; originals and three iterations of enhancement

```
      20   19   18   18   17   18   19   20   20   20   19   18   18   19   19   19   19   19   19   19
      20   20   19   18   19   19   20   20   20   20   19   19   19   19   19   19   18   18   19   20
      20   20   20   20   21   21   20   20   20   20   20   20   19   19   19   19   18   18   19   19
      20   20   21   22   22   21   21   20   20   21   21   20   20   20   20   19   19   19   19   19
      21   21   22   22   23   22   21   21   21   22   22   22   21   20   21   20   20   20   21   20
      24   23   23   24   24   22   23   23   24   24   25   24   23   22   23   23   23   23   24   23
      28   28   28   28   28   28   29   29   30   30   30   30   29   29   29   29   29   30   30   30
      34   34   34   35   35   35   35   36   36   36   36   36   36   36   36   36   36   36   36   36
      38   38   38   38   38   38   38   38   39   39   38   39   38   38   38   38   38   38   38   38
(d1)  36   36   36   36   36   36   36   36   36   35   35   35   35   35   35   35   35   35   34   34
      30   30   31   30   29   29   29   29   29   28   28   28   29   29   29   29   29   29   28   27
      25   26   26   25   25   25   24   25   24   23   23   24   24   25   26   25   26   25   24   24
      24   23   23   23   23   23   23   23   22   22   22   22   23   24   24   23   23   23   22   23
      23   22   22   21   22   21   21   21   21   21   22   22   23   23   22   22   22   22   22   22
      22   21   21   20   20   20   20   20   21   21   21   22   22   22   21   21   21   21   21   21
      21   20   21   20   20   20   20   20   20   21   21   21   21   21   20   20   20   20   21   21
      20   20   20   20   20   20   20   20   20   20   20   20   20   20   20   20   20   20   21   20
      20   20   19   19   19   19   20   20   20   19   19   19   19   19   19   19   20   20   20   19
      20   19   18   18   18   19   19   19   19   19   19   19   19   19   19   20   20   19   19   19
      20   19   19   18   18   19   18   18   18   19   19   19   19   19   19   20   19   19   19   19
```

```
       1    2    2    2    2    2    2    1    1    2    2    2    2    1    1    1    1    0    1    2
       1    1    2    3    4    3    2    1    0    1    1    2    1    1    0    1    1    0    1    0
       1    1    2    4    3    2    1    1    1    1    2    2    1    1    1    1    1    1    1    0
       1    2    2    2    2    2    1    1    2    2    2    2    2    2    2    2    2    2    2    2
       4    3    3    2    2    2    3    3    4    4    4    4    3    3    3    4    4    5    5    5
       7    7    7    6    6    7    8    9    9    9    8    8    9    9    9    9   10   10   10   10
      11   11   11   11   12   12   13   13   13   12   12   12   13   14   14   13   13   13   13   13
      10   10   10   10   10   10   10    9    9    9    9    9    9    9    9    9    9    9    8    8
       3    2    2    2    1    1    1    1    0    1    1    1    1    1    1    1    1    1    1    2
(d2)   7    7    7    8    8    9    9    9   10   10   10   10    9    9    9    9    9    9   10   10
      10   10   10   10   11   11   11   11   11   12   11   11   10   10    9    9    9    9   10   10
       6    7    7    7    6    6    6    6    6    6    6    6    5    5    5    5    6    6    5    4
       2    3    4    3    3    3    3    3    3    2    1    1    1    2    3    3    3    3    2    2
       2    2    2    2    3    3    3    2    1    1    0    1    1    2    2    2    2    1    1    1
       2    1    1    1    1    1    1    1    0    0    0    1    1    2    2    2    2    1    1    1
       1    1    1    1    0    0    0    0    0    1    1    1    2    1    1    1    1    0    0    1
       1    1    1    1    1    0    0    0    0    1    2    2    2    1    1    0    0    0    1    1
       2    1    1    2    1    1    1    1    1    1    1    1    1    1    0    0    0    1    1    1
       2    2    1    1    0    0    1    2    1    0    0    0    0    0    1    1    1    1    1    1
       2    2    1    1    1    1    1    0    0    0    0    0    0    0    0    0    1    1    1    1
```

```
       0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
       0    0    0    7    8    7    0    0    0    0    0    0    0    0    0    0    0    0    0    0
       0    0    0    7    8    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
       0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
      10    0    0    0    0    0    0    0    0   12   12   11   10    0    0    0    0    0    0   14
      19   19   17   17   16   19   21   23   23   22   21   22   22   23   23   24   25   25   25   25
      26   26   26   26   26   28   28   30   29   28   27   28   29   30   30   30   30   30   29   29
      22   22   22   22   22   23   22   22   21   20   19   20   20   20   20   20   20   20   19   18
       0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
(d3)  14   14   15   16   17   18   19   19   20   22   21   20   20   19   19   19   18   19   20   21
      22   22   23   23   23   24   25   24   26   25   26   24   24   21   21   22   21   22   22   22
      17   17   18   18   17   17   17   17   17   16   16   15   15   13   14   15   16   15   14   13
       0    9   10   10    0    9   10   10    8    0    0    0    0    0    8    9    9    0    0    0
       0    0    0    0    5    6    6    0    0    0    0    0    0    0    0    0    0    0    0    0
       0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
       0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
       0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
       0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
       0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
       0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
       0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
```

Figure 1. (d) Gradient magnitudes displayed numerically on a scale of 0-63 for a small subwindow, indicated by tick marks in (a): (d1) subwindow gray levels; (d2) original gradient magnitudes; (d3-5) results of three iterations

(d4)

```
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0  12   8   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0  31  28   0   0   0   0   0   0   0   0
  46  45  41  40  42  42  47  49  55  52  54  53  54  54  51  52  53  53  57  58
  61  62  61  60  63  63  63  63  63  63  63  63  63  63  63  63  63  63  63  63
  48  47  47  46  48  48  50  47  47  45  45  45  46  47  47  47  47  46  45  45
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  30  35  34  36  39  41  41  42  43  45  45  44  41  41  40  38  39  39  42  43
  49  51  51  53  54  56  56  58  57  60  58  58  53  52  49  50  51  51  53  48
  38  40  45  43  45  41  44  46  43  42  36  37  34  37  35  40  39  39  32   0
   0   0  24   0   0   0  26  26   0   0   0   0   0   0   0  21   0   0   0   0
   0   0   0   0   0  11   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
```

(d5)

```
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   8   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  63  63  63  63  63  63  63  63  63  63  63  63  63  63  63  63  63  63  63  63
  63  63  63  63  63  63  63  63  63  63  63  63  63  63  63  63  63  63  63  63
  63  63  63  63  63  63  63  63  63  63  63  63  63  63  63  63  63  63  63  63
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  63  63  63  63  63  63  63  63  63  63  63  63  63  63  63  63  63  63  63  63
  63  63  63  63  63  63  63  63  63  63  63  63  63  63  63  63  63  63  63  63
  63  63  63  63  63  63  63  63  63  63  63  63  63  63  63  63  63  63   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
```

Figure 1(d), continued

(e1)

```
 225 207 207 243 270 297 297 315 270 207 225 243 270 315 270 180 180     270 270
 225 225 225 270 270 270 270 270     225 225 243 270 270     180 180       0
 180 270 270 270 270 243 225 225 270 270 243 243 225 270 225 225 225 270   0
 225 270 297 297 243 225 225 225 270 270 243 243 243 270 243 243 243 270 270 243
 256 270 288 297 243 225 252 270 284 284 256 243 236 270 270 256 270 270 270 259
 262 262 270 270 260 262 270 270 277 270 270 256 257 270 270 263 270 276 270 264
 264 264 270 270 266 270 270 274 270 270 270 266 266 270 270 270 270 270 270 266
 270 270 270 270 270 270 270 277 270 263 270 263 263 270 270 270 270 270 270 270
 252 270 270 270 270 270 270 270     180  90  90 135  90  90  90  90 135 135  90
  93  90  90  97  97  90  90  90  96  96  90  90  90  90  90  90  90  97  96  90
 101  90  96 101  96  96  90  90 100  94  90  90  90  90  90  90  90 103 101  90
 100  90  98  98 100 100  90 100 108 100  90  80  79  79 101 101  90 108 101  90
 117 108 104 108  90 108  90 108 108  90  90  45  45  90 108 108  90 108 117  90
 117 117 117 117  90 108  90  90  90  90       0  45 117 117 117  90 135  90  90
 117 135 135 135  90 135  90  90              90  90 117 117  90  90  90  90 135
 135 135 180 180                      90  90  90  90 135 135  90  90         180
 180 180 135 135  90                  90  90  90  90 135 135              90 135
 180 135 135  90  90  90  90  90 135 135  90  90  90  90              90 135 135
 180 180 180 180          90  90  90                     270 270 180 135 180 180
 180 207 225 225 270 270 180                                     180 225 225 225
```

Figure 1. (e) Gradient directions displayed in degrees for the subwindow: original

```
                   267 273 276
                   273 270
       260                                 278 262 252 249                          264
(e2)   264 266 270 269 266 266 271 271 276 271 269 262 262 269 271 270 271 273 270 269
       267 269 270 270 269 271 273 274 270 271 267 269 269 269 270 271 271 271 271 270
       269 270 270 271 271 271 271 274 270 269 270 267 269 270 270 270 270 270 270 271

        98  94  91  93  94  90  90  91  93  93  91  90  90  91  90  90  91  93  93  93
        97  91  93  96  94  93  90  91  94  93  90  89  89  90  90  89  91  97  97  91
        96  91  96  97  98  94  91  96 100  97   9  84  82  82  93  94  93 101 100  91
           105 100  98     100  96 103 103                     101 101 100
                        98  96  96
```

```
                       280 287
                                                   266 262
       264 264 270 270 269 269 271 271 277 274 270 264 263 266 271 271 271 273 274 274
(e3)   269 269 270 270 270 273 273 274 271 271 269 269 270 270 271 271 271 271 271 270
       271 270 270 271 271 271 273 273 271 269 270 270 270 271 270 271 271 271 270 271

        98  91  90  91  91  89  90  90  90  91  90  90  90  90  89  90  90  91  91  91
        94  91  91  94  93  91  90  91  93  91  90  89  89  89  89  89  91  94  94  96
        90  89  94  97  96  89  91  94  98  96  89  86  84  82  87  91  96  98  94
                96              97 100                              96
                           105
```

```
                           270

       264 269 271 270 270 271 273 271 274 276 274 264 264 270 271 271 271 271 273 276
(e4)   270 270 270 271 271 273 273 274 273 273 270 270 270 270 271 271 271 273 273 271
       271 270 270 271 271 271 273 273 271 270 270 270 270 271 271 271 271 271 271 271

        94  90  90  90  90  89  90  90  90  90  90  90  90  90  89  89  90  90  91  91
        93  90  90  93  91  91  89  91  91  91  90  89  87  87  87  89  91  93  97  96
        91  87  91  97  93  86  87  96  97  91  89  87  86  86  84  90  96  93
```

Figure 1. (e) Gradient directions displayed in degrees for the subwindow: three iterations

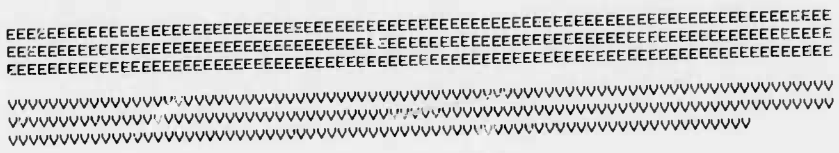


Figure 2. (a) Edge components extracted from the subwindow in Figure 1

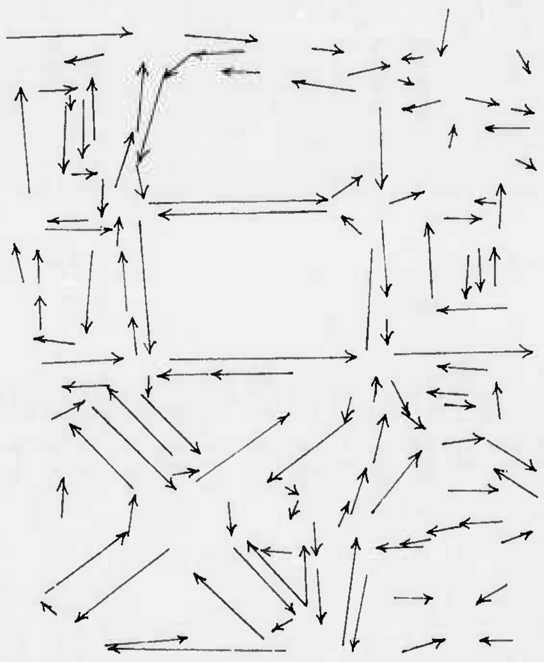


Figure 2. (b) Line segments fitted to the edge components in the window of Figure 1

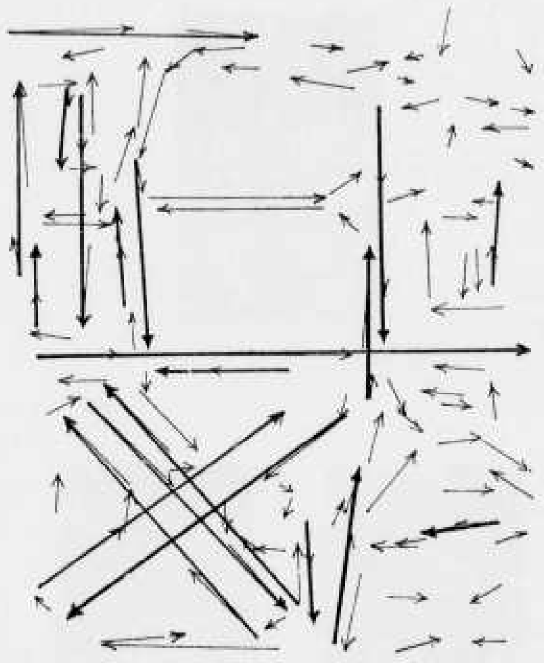


Figure 3. Results of collinear linking (heavy lines) for the window of Figure 1

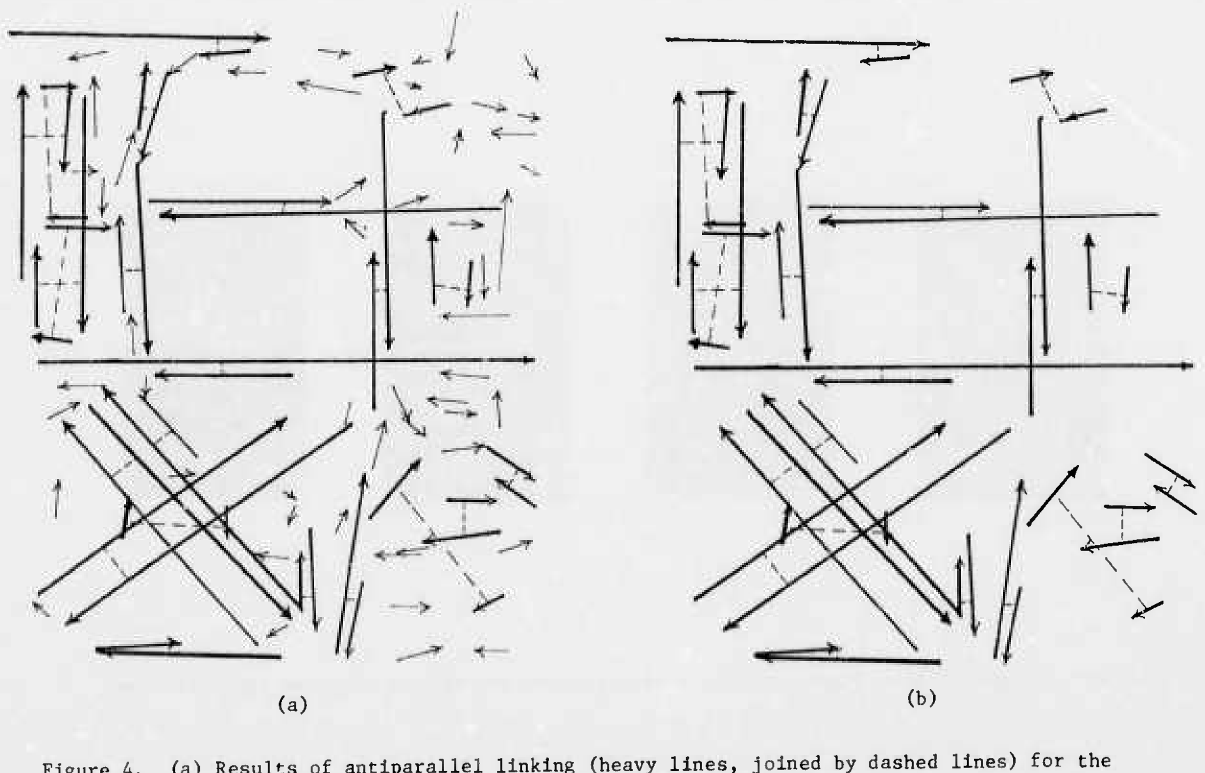
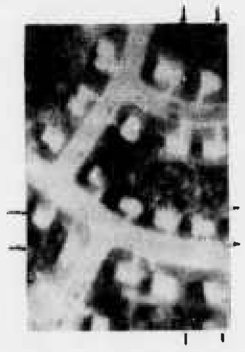
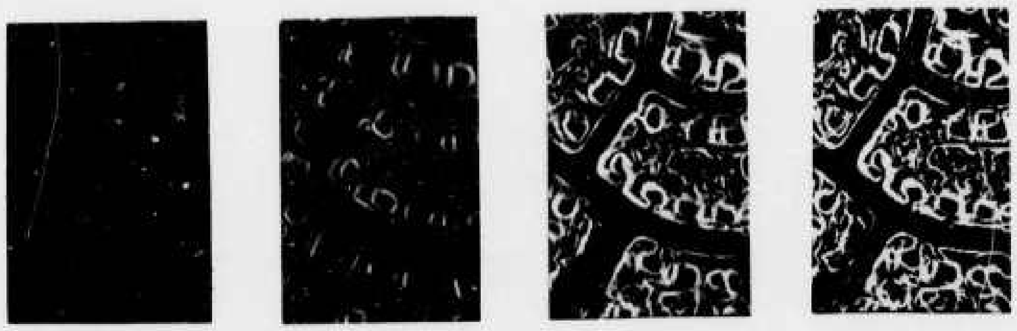


Figure 4. (a) Results of antiparallel linking (heavy lines, joined by dashed lines) for the window of Figure 3. (b) The antiparallel pairs only

(a)



(b)



(c)

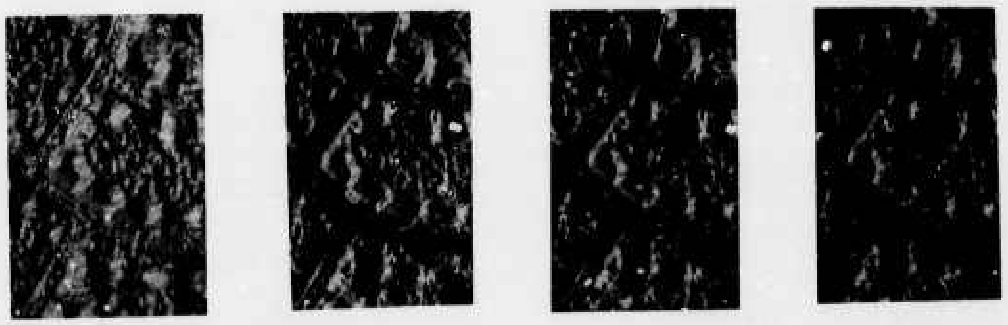


Figure 1'. Analogous to Figure 1 for a second window showing part of a new suburban area

(d1)

```
 28  27  26  28  29  29  26  21  20  20  22  25  28  29  29  28  26  23  22  22  24  24  25  26
 27  25  23  23  25  27  26  24  23  24  25  27  28  29  28  27  25  23  23  24  23  23  24  26
 24  22  21  23  25  27  30  31  30  30  29  28  27  27  26  24  23  22  23  25  25  24  24  25
 21  21  23  26  28  31  34  35  35  35  33  29  27  26  26  24  23  22  23  25  26  25  25  26
 19  21  26  31  34  35  36  37  38  37  35  31  29  28  28  27  25  24  24  26  27  27  25  26
 19  21  27  31  35  37  38  38  39  39  38  34  32  31  32  31  29  28  27  28  29  28  26  27
 19  20  22  27  33  38  39  40  40  41  40  37  35  35  36  36  34  31  30  30  30  29  26  24
 19  18  19  23  32  39  41  41  41  41  41  40  40  39  40  39  38  36  33  31  30  28  23  18
 18  19  19  23  33  40  42  42  42  42  42  42  42  42  42  41  41  40  37  33  30  26  19  14
 17  18  19  23  33  40  42  43  43  43  43  43  43  43  43  42  42  41  39  35  30  25  18  13
 14  16  18  24  34  41  43  43  43  43  43  43  43  43  43  43  42  42  40  36  29  25  18  13
 12  13  18  25  35  41  43  43  44  43  43  43  44  44  43  43  43  42  40  36  30  25  18  13
 12  13  18  25  36  42  43  43  43  44  44  44  44  44  43  43  43  42  41  36  31  26  19  14
 15  15  18  25  35  42  43  43  43  44  43  43  43  43  43  43  43  42  40  37  31  26  20  17
 19  20  20  25  35  41  43  43  43  43  43  43  43  43  43  43  42  42  40  36  30  26  22  20
 24  24  23  27  35  39  42  42  42  42  42  42  42  43  42  42  42  40  39  34  27  26  24  22
 27  27  27  28  31  35  39  39  39  38  39  39  40  41  40  40  39  38  36  33  29  26  25  24
 29  29  28  29  31  34  36  35  34  34  34  36  36  37  37  36  34  33  32  31  29  28  25  24
 30  30  31  34  36  37  36  33  31  31  32  33  33  34  35  35  32  30  30  30  29  28  25  23
 34  34  36  39  40  40  37  34  32  33  33  33  32  34  36  36  33  31  30  30  30  28  25  23
 39  40  40  41  42  42  40  38  36  36  36  36  36  37  38  38  36  34  33  33  33  32  29  28
 42  42  43  43  43  43  42  41  40  40  40  40  40  40  40  40  39  39  38  38  38  37  36  35
 40  42  43  43  44  43  43  42  42  42  42  42  42  42  41  41  41  41  41  41  41  40  39  38
 38  40  42  43  43  43  43  43  43  43  42  43  43  43  42  42  42  42  42  42  42  42  41  39
```

(d2)

```
  1   2   4   6   5   4   7   6   4   5   6   6   4   1   1   4   6   4   3   3   1   1   2   1
  3   5   5   4   4   1   5   9  10  10   7   3   1   2   3   4   4   2   1   2   2   0   2   2
  5   3   1   4   5   6   8  11  12  11   8   3   1   2   3   3   3   1   2   2   3   2   1   2
  3   2   7   9   9   8   7   7   8   7   7   6   3   2   2   4   4   2   2   2   2   3   2   2
  1 [illegible] 8 9 8 6   5   4   4   5   7   7   5   6   6   7   7   6   5   4   3   3   2   2
  2   6   9   8   6   3   3   3   3   4   7   8   7   7   8   9   9   8   6   5   3   3   2   2
  2   4   9  11  11   6   3   3   3   3   5   7   8   8   8   9   9   8   6   4   2   5   6   8
  2   1   5  12  15   8   3   3   2   2   3   5   7   7   6   6   7   9   8   5   4   8  11  11
  3   1   4  13  16   9   2   2   2   2   3   3   4   4   4   4   4   6   9   8   7  11  12   8
  4   2   5  14  17   9   2   1   1   1   1   1   1   1   2   2   2   3   7   9  10  12  12   7
  4   5   8  15  16   8   2   1   1   1   0   1   1   1   1   1   1   3   6  11  11  12  12   7
  1   5  11  17  16   8   1   0   1   1   1   1   1   1   1   1   1   3   6  11  11  12  12   7
  3   5  11  17  16   7   1   0   0   1   1   0   0   1   1   0   1   3   6  10  11  12  12   8
  7   6   9  16  16   7   1   0   0   0   1   1   1   1   1   1   1   3   6  10  11  11   9   7
  9   8   7  14  15   7   2   1   1   1   1   1   0   0   1   1   2   3   6  10  10   8   8   7
  8   8   6  10  11   8   5   4   4   4   4   3   3   2   2   3   3   4   6  10   9   6   5   5
  5   5   5   5   7   8   6   7   7   8   7   6   6   5   5   6   7   7   7   7   7   5   3   2
  3   4   5   6   7   4   2   5   7   7   6   6   6   6   5   5   7   7   5   5   4   4   4   2
  6   6   8  10   9   6   4   5   2   1   1   2   3   2   1   3   5   3   1   2   3   5   5   2
 10  10   9   8   6   5   7   7   5   5   4   4   3   4   3   5   6   5   4   4   4   6   7   5
  8   8   7   5   4   4   6   8   8   8   7   8   8   6   5   5   7   8   8   8   9  10  11  12
  2   2   3   3   2   2   4   5   6   6   6   6   6   5   4   4   5   7   8   8   8   9  10  10
  4   2   1   0   0   1   1   2   3   3   3   3   3   3   3   3   3   4   4   4   5   5   5   6
  3   3   2   1   0   1   1   1   1   1   1   1   1   2   2   2   2   2   2   2   2   2   4   6
```

(d3)

```
  0   0   0  11   8   0   0  13  10  13  16  15  10   0   0   9  11   8   6   0   0   0   0   0
  6  10   8   7   0   0   0  19  24  23  16  10   0   0   7   9   8   0   0   0   0   0   0   0
  9   8 [illegible] 0 0  12  18  24  28  25  16  10   0   0   4   6   5   0   0   0   5   0   0   0
  6   0   0  19  19  20  17  19  20  19  19  14   9   0   0   0   0   0   0   0   0   6   0   0
  0   0 [illegible] 0 14  15  12  12  12  15  18  18  15  12  15  15  16  14  11   9   8   5   0   0
[illegible] 0 20 20 16 12   8   8   8  11  15  19  17  18  20  21  20  18  15  10   8   8   0   0
  0   0  20  26  23  15   9   8   7   8  13  17  18  20  20  20  22  21  15   9   0  11   0  20
  0   0  15  28  32  21  10   6   0   0   9  12  16  17  16  15  18  21  19  14   0  18  24  25
  6   0   0  31  37  24   0   0   0   0   5   8   9  11  10   9  11  17  20  19   0  27  28  22
 10   0   0  33  36  23   0   0   0   0   0   0   0   0   0   0   0  11  18  22  27  30  29  19
  9  13   0  37  38  23   0   0   0   0   0   0   0   0   0   0   0   8  17  26  30  31  28  19
  0  12   0  39  36  21   0   0   0   0   0   0   0   0   0   0   0   8  17  25  30  32  29  19
  7  13  29  41  38  21   0   0   0   0   0   0   0   0   0   0   0   8  17  25  29  30  27  18
 16  17  22  38  37   0   0   0   0   0   0   0   0   0   0   0   0   8  16  24  28  27  24  20
 21  20   0  31  33  20   0   0   0   0   0   0   0   0   0   0   0   9  16  24  26  22  18  16
 20  18  17  22  26  20  13   9  11   0   0   0   7   0   0   7  10  13  17  22  21  16   0  10
 13  14   0  15  16   0  12  14  17  17  16  14  13  12  11  12  14  16  18  17  17  12   9   0
 10   0  14  14   0   0   0   0  14  14  14  14  13  12  10  11  12  14  13   0  11  12   9   0
 17  16  19  21  18   0   0   0   0   0   0   0   0   0   0   0  10   8   0   0   8  11  11   0
 23  22  22  19  16  13  16  16  12  10   0   8   8   6   7   8  11  10  10  10  12   0  18  14
 18  18  16  13  10  10  15  17  18  17  17  16  16  14  10  11  16  17  18  19  19  21  24  24
  0   0   0   6   0   0   0  12  15  15  15  15  15  14  10  10  13  16  18  19  19  21  23  21
  5   0   0   0   0   0   0   0   0   9   0   0   9   8   7   7   9  10  12  13  12  13  15  17
  5   6   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0  10  14
```

Figure 1'. (d1-3)

(d4)

```
   0   0   0   0  10   0   0  28  24  29  40  37  23   0   0  19  22   0  10   0   0   0   0   0
   0  13  18   0   0   0   0   0  55  54  40  21   0   0   0  15  17   0   0   0   0   0   0   0
  24  15   0   0   0   0  41  52  63  59  40  20   0   0   0   0   0   0   0   0   0   0   0   0
  18   0   0   0  29  38  44  47  51  48  46  38  22   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0  33  36  35  32  33  39  47  45  35  34  31  34  31  28  23  20   0   0   0   0
   0   0  36  30  32  30  26  20  22  29  43  46  42  47  49  52  50  44  34  24  18   0   0   0
   0   0  34  55  58  40  26  18  16  19  32  41  45  49  49  51  51  47  38   0   0  21   0  43
   0   0   0  63  63  51  24   0   0   0  18  29  39  42  40  39  45  52  49   0   0  30  50  62
   0   0   0  63  63  58   0   0   0   0   0   0  23  24  23  22  30  44  51  47   0  62  62  55
  18   0   0  63  63  57   0   0   0   0   0   0   0   0   0   0   0  29  47  54   0  63  63  48
   0   0   0  63  63  53   0   0   0   0   0   0   0   0   0   0   0  20  43  63  63  63  63  51
   0  31   0  63  63  54   0   0   0   0   0   0   0   0   0   0   0  21  45  63  63  63  63  46
  19  21   0  63  63   0   0   0   0   0   0   0   0   0   0   0   0  21  43  62  63  63  63  48
  42  44  51  63  63   0   0   0   0   0   0   0   0   0   0   0   0  21  43  61  63  63  55  45
  52  35   0  63  63  33   0   0   0   0   0   0   0   0   0   0   0  24  43  60  63  50  38  41
  47  34  35  46  52  39  31  22   0   0   0   0   0   0   0   0   0  32  42  55  55  42   0   0
  33   0   0  32   0   0   0  33  29  36  31  29  25  25  25  28  32  34  41  35  37  32   0   0
  21   0  23  29   0   0   0   0   0  29  28  27  26  24  24  21  25  32  27   0  26  29  20   0
  38  43  42  40   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0  10  25  28   0
  54  53  49  43  33  34  23  21  22   0   0   0  18  18   0  17  21  26  24  25   0   0  34  34
  41  39  37  28  22  20  32  41  39  38  37  35  35  30  27  25  29  37  41  43  45  49  45  56
   0   0   0   0   0   0   0   0  34  32  34  35  33  31  27  25  30  40  44  45  48  51  54  50
   0   0   0   0   0   0   0   0   0   0   0   0   0  19  17  17  19  24  27  28  29  34  38  37
   0   4   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0  27  37
```

(d5)

```
   0   0   0   0   0   0   0   0  54  63  63  63  50   0   0  37  48   0   0   0   0   0   0   0
   0   0  17   0   0   0   0   0   0  63  63  51   0   0   0   0   0   0   0   0   0   0   0   0
  44  35   0   0   0   0   0  63  63  63  63  49   0   0   0   0   0   0   0   0   0   0   0   0
  41   0   0   0   0  63  63  63  63  63  63  63  54   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0  63  63  63  63  63  63  63  63  63  63  63  63  61  51   0   0   0   0   0
   0   0   0   0  63  63  63  58  56  63  63  63  63  63  63  63  63  63  63  46  31   0   0   0
   0   0   0  63  63  63  63  46  29  47  63  63  63  63  63  63  63  63   0   0   0  29   0   0
   0   0   0  63  63  63  53   0   0   0   0  63  63  63  63  63  63  63  63   0   0  63  63  63
   0   0   0  63  63  63   0   0   0   0   0   0   0  57  55  53  63  63  63   0   0  63  63  63
   0   0   0  63  63  63   0   0   0   0   0   0   0   0   0   0   0  63  63  63   0  63  63  63
   0   0   0  63  63  63   0   0   0   0   0   0   0   0   0   0   0  52  63  63  63  63  63  63
   0   0   0  63  63   0   0   0   0   0   0   0   0   0   0   0   0  53  63  63  63  63  63  63
  38  59   0  63  63   0   0   0   0   0   0   0   0   0   0   0   0  55  63  63  63  63  63  63
  63  63  63  63  63   0   0   0   0   0   0   0   0   0   0   0   0  54  63  63  63  63  63  63
  63  63   0  63  63  63   0   0   0   0   0   0   0   0   0   0   0  60  63  63  63  63   0   0
  63  63  54  63   0   0  63   0   0   0   0   0   0   0   0   0  46  63  63  63  63  63   0   0
  48   0   0  63   0   0   0   0  63  50  63  55  52  48  47  62  63  63  63  63  63  63   0   0
   0   0  59   0   0   0   0   0   0   0  60  54  51  52  50  52  56  58   0   0  54  63   0   0
  63  63  63   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0  56  63   0
  63  63  63  63  63  59  60  49   0   0   0   0   0   0   0   0   0  39  54   0   0   0  63  63
  63  63  63  62  49  49  51  63  63  63  63  63  63  63  62  52  63  63  63  63  63  63  63  63
   0   0   0   0   0   0   0   0   0  63  63  63  63  63  63  63  63  63  63  63  63  63  63   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0  42  42  48  56  63  63  63  63  63  63
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0  57  63
```

Figure 1'. (d4-5)

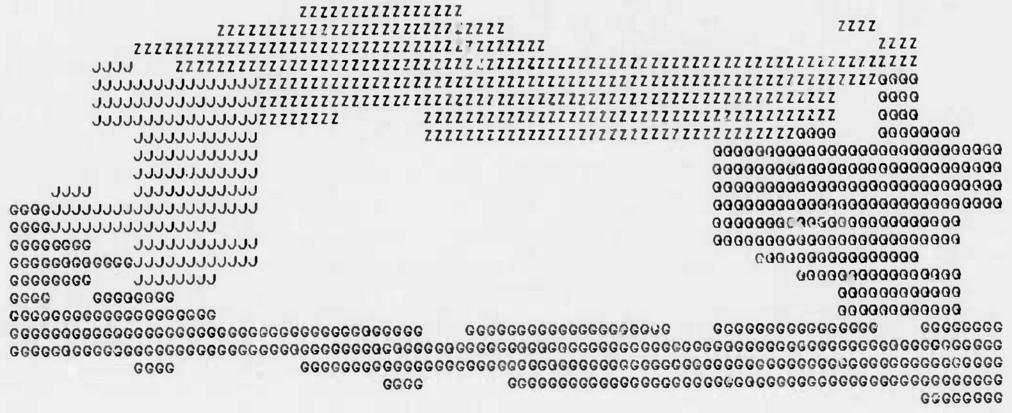
(e1)

```
 315 135  90  72  79 135 180 198 243 281 308 322 315 315 180 180 190 207 252 288 315  45  45  45
 108 135 101  63  45  45 233 252 264 276 285 304   0  90 124 135 153 153 315 297 243       0   0
 113 124   0 315 315 308 284 264 266 264 256 236 180 117 146 146 146 135   0 297 252 243 315 333
 108   0 315 302 294 297 293 278 263 255 231 218 214 243 225 225 225 243 315 315 270 236 270 333
  90 350 333 319 300 297 293 284 270 239 219 219 239 260 260 248 248 260 281 277 270 236 243 333
  27  10  18  21   0 326 288 288 270 243 225 225 248 270 270 252 246 249 270 281 270 214 207  90
  27  45  41  31  15 350 288 270 270 252 225 231 256 270 270 252 246 243 252 256 225 180 153 117
  27  45  23  10   0 346 304 270 270 270 236 248 262 270 270 252 248 238 235 225 180 159 149 128
  18   0   0 356 356 353 315 270 270 270 252 252 256 270 256 243 243 232 221 210 188 170 162 145
  45  45  11 356 357 353 333 270 270 270 270 270 270 270 243 243 243 214 203 193 180 176 176 172
  63  45   7 353 353 346 333 270 270 225     270 270 225 225 225 225 198 190 186 186 180 180 180
  45  23   0 357 353 353 315     270 225 270 270 270 225 225 225 225 198 190 186 190 190 184 188
 270 338 354   0 356 352 315         270 225         180 180     180 180 190 186 186 190 194 210
 270 297 342 356   0   0   0             135  90  90 180 180 180 180 180 180 180 180 186 198 231
 270 277 309 347   0   8  27  90  90  90 135  90         180 135 153 162 170 169 174 187 210 236
 277 263 297 343  10  35  53  90 104  90  90  90  72  90 117 108 124 135 153 169 173 190 217 233
 281 259 281 338  15  35  63  98  98  90  82  80  80  90 101 108 113 113 129 156 180 203 214 225
 304 256 281 297 315 333  90 121 105  90  80  80  80  80 101 121 124 113 121 143 180 194 194 180
 280 270 291 293 288 270 207 180 153  90  45  90  72  45  90 180 169 162 135 180 198 191 180 180
 270 276 288 291 280 248 225 225 248 270 270 256 288 315 288 233 218 233 256 256 243 218 225 248
 270 277 285 293 270 243 232 240 256 270 270 263 270 288 281 248 236 249 263 270 257 248 250 256
 270 297 288 270 270 225 225 239 260 270 270 270 270 270 270 256 248 255 263 270 263 252 253 253
  63  45  45         225 225 243 252 252 270 270 270 252 252 252 252 256 256 270 259 248 239 218
  34  34  27  45     180 225 225 270 225 270 270 270 243 243 270 270 270 270 243 243 243 225 198
```

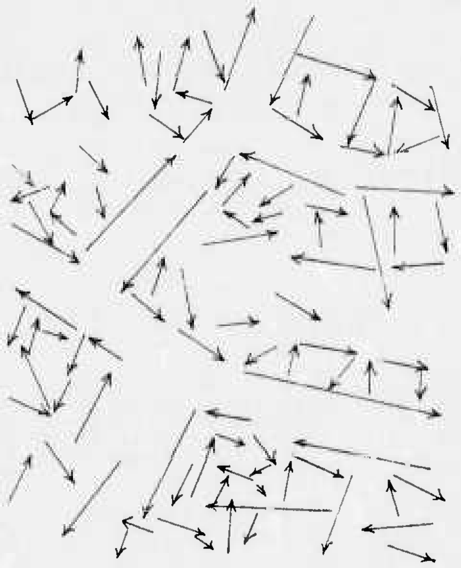
Figure 1'. (e1)

```
                    80  87         211 239 285 308 316 316         174 186 205 248
       114 125 107  80             243 266 276 294 304         125 139 169
       121 122             294 284 270 269 266 249 240         160 166 172             257
       124         307 293 301 290 280 264 245 231 224 226                                 235
               346     293 288 287 281 267 236 222 222 238 257 260 256 255 263 276 287 276 256
                13  11 354 329 304 283 267 240 226 228 253 267 266 259 256 257 267 274 264 212
                38  31   6 343 305 277 273 256 228 232 259 270 270 259 242 239 243 246     183     122
                27  11   0 350 307 276         236 242 263 267 267 259 242 236 232 224     170 145 129
        25           0 359 354                 267 267 266 266 259 255 250 231 222 214     172 165 143
        41         356 356 354                                             218 208 203 181 177 176 170
        55  42     354 356 352                                             198 191 187 184 181 181 181
            25     357 356 353                                             193 187 186 187 187 186 188
       284 328 356   0 359 356                                             183 186 184 186 186 190 215
       276 305 333 359   0                                                 177 179 180 180 184 195 226
(e2)   273 281     353   3  13                                             163 172 173 176 186 214 235
       274 270 304 346   8  34  55  82  96              79         120 128 135 146 170 177 187     245
       276 267     336   3      68  87  91  71  86  82  83  89  98 103 104 118 131 153 181 197 201
       304     277 288                 101  90  82  79  83  86 100 108 110 110 122     187 197 193
       277 280 285 287 283                                             173 165         204 201 193
       271 278 285 298 276 252 229 226 252 267     263 285 295 278 246 207 245 255 257 248     225 252
       271 274 278 283 269 252 233 238 259 267 269 269 273 280 276 257 238 245 263 267 260 256 257 260
                   269             238 260 267 269 271 269 267 266 260 257 260 266 267 264 260 259 257
        75                                 262         266 257 257 259 263 264 263 267 262 255 233 221
        21  38                                                                                 222 211

                       103         203 250 295 308 315 315         170 183     235
           110 110                     267 277 295 304             159 174
       124 128                 283 277 270 266 248 242
       127             278 291 287 280 266 243 231 226 228
                       312 302 287 280 264 235 225 225 238 256 264 263 263 269 276 280
                17  23 350 328 308 295 266 238 226 229 253 264 267 262 260 260 266 276 262
                18  18   1 343 311 290 276 252 231 233 259 269 270 262 242 239 240         191     128
                    13   1 350 309             232 255 264 269 267 260 240 233 231         188 139 131
                     4 359 353                         269 267 263 260 243 229 224 214     170 155 143
        31         359 357 357                                             217 212 207     173 176 169
(e3)               359 356 356                                             195 194 187 184 181 181 181
            23     354 357 357                                             190 186 184 186 186 186 191
       288 346     356 356                                                 183 184 183 184 184 190 215
       278 307 328   3   0                                                 179 179 180 180 184 194 226
       274 300     354   8  30                                             153 172 174 177 180 217 233
       273 269 300 352   0  38  58  80                                 121 134 145 170 177 188
       280         338              80  77  86  87  89  86  86 100 103 104 111 132 146 188 195
       309     300 308                      89  84  83  84  87  93 104 105 108 129     190 197 190
       274 281 287 277                                                                 180 215 204
       271 277 283 285 267 255 250 250 257             280 285     245 201 233 255 260         224 256
       271 274 274 276 274 259 238 238 257 263 269 274 271 277 274 257 252 256 264 266 262 259 267 263
                                       259 267 273 267 266 267 266 263 262 263 266 267 266 263 260 260
                                                           262 262 266 270 270 269 269 266 257 235 225
            53                                                                                 218 215

                                       242 297 309 314 314         176 184
               114                         278 297 304
       122 124                     280 277 267 246 242
       129                 295 288 280 269 242 232 229 231
                           307 288 280 263 235 226 226 239 256 266 266 267 271 277
                       349 328 311 298 264 238 228 231 255 264 267 264 263 262 271 270 249
                    21   0 343 315 293 271 252 231 235 260 269 270 263 242 238             195
                     8   1 352 318                 246 263 269 269 260 239 233 225         188 138 131
                     0   0 354                             269 266 262 240 229 224         176 149 142
                     0 359 359                                             217 213 208     170 176 166
(e4)                 0 357 359                                             195 194 194 205 176 181 181
                   359 353                                                 188 184 184 184 164 186 193
       300 349     352 353                                                 183 163 183 184 184 188 215
       281 208 308   0   3                                                 179 179 180 180 183 205 224
       277 305     354  10  35                                             152 172 176 179 177
       277 269 297 350          62                                     124 131 146 158 179 187
       298         339                  75  80  89  89  87  89  94 101 104 110 134 145 191 195
               309                              84  84  86  89  93  98 100 104         190 205
       271 281 287                                                                         217 210
       271 276 278 274 264 262 255 255                                     248 260             226 259
       271 273 273 271 270 262 255 252 256 263 269 276 276 270 271 262 257 260 264 263 262 264 271 273
                                           267 270 270 266 264 267 266 266 267 267 269 267 264 263
                                                               264 269 273 273 270 270 269 257 235 224
                                                                                               224 219
```

Figure 1'. (e2-4)



(a)



(b)

Figure 2'. Analogous to Figure 2 for the second (sub)window

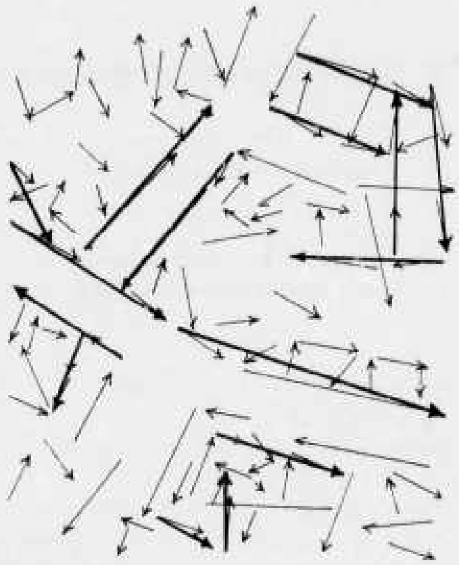
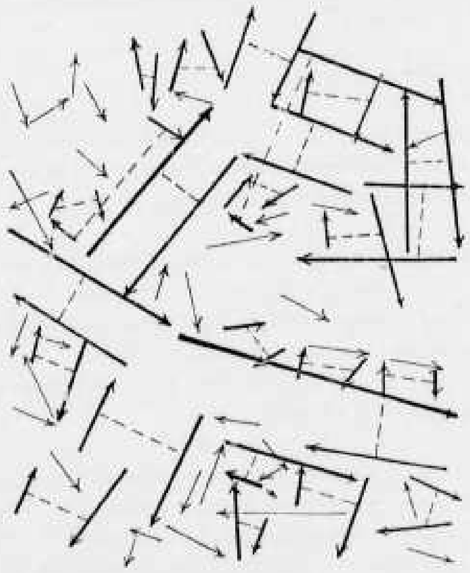
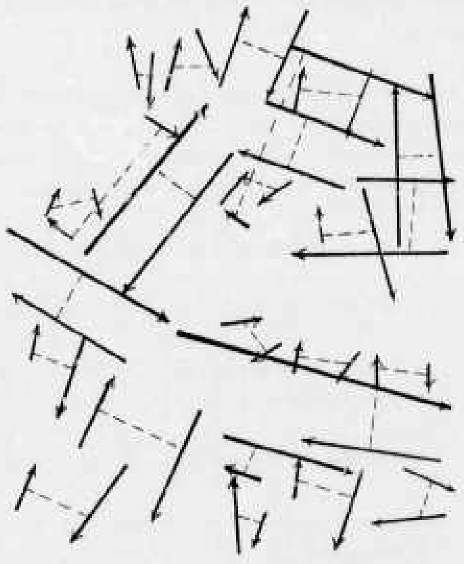


Figure 3'. Analogous to Figure 3 for the window of Figure 1'.



(a)



(b)

Figure 4'. Analogous to Figure 4 for the window of Figure 3'.

MIT'S REPRESENTATION TECHNIQUES

Patrick H. Winston and the Staff

The Artificial Intelligence Laboratory
Massachusetts Institute of Technology

In this series of image understanding conference proceedings, we have stressed the key issue of representation. In particular, we have described the work of Horn and his collaborators using the reflectance map *and the* albedo image *in working with satellite images, and we have described the work of Marr and his collaborators using the* primal sketch, *the* 2 1/2-D sketch, *and* body-centered, 3-D models *to work toward a comprehensive theory of recognition.*

Some of this material has be adapted from previous workshop proceedings.

Horn Concentrates on Representing Image-formation Constraints

Understanding an image implies a need to understand how light reflection depends on various combinations of surface material, surface orientation, and light-source position. Among the products are tools for dealing with the following needs:

* Automated generation of shaded relief maps.

* Generation of low-level, obliquely-viewed images.

* Generation of special maps that bring out particular terrain features.

* Classification of ground cover for crop prediction.

* Matching images to terrain data for satellite navigation.

* Making maps for automatic or semiautomatic change detection.

Doing all this requires a number of key representations: the reflectance map, the digital terrain map, the synthetic image, the multiple-sun synthetic image, the albedo image, and the change-detection image. Since understanding reflectance maps is prerequisite to following Horn's work, we now describe what is involved.

The Reflectance Map

The purpose of the reflectance map is to make explicit the relationship among observed intensity, surface material, surface orientation, and light-source position. To see how, consider figure 1. All points (p, q) in the space correspond to surface orientations. For a given surface material and light-source position, a surface's orientation determines its reflected light intensity. By drawing lines through points representing orientations that have the same intensity, one gets the isointensity lines shown. This particular map is for illumination from the upper left.

Synthetic Images

Once it is possible to predict intensities from material, orientation, and light-position information, it is then possible to produce synthetic high-altitude images.

Figure 2 shows an image of a piece of Switzerland synthetically generated using a digital terrain model and a simple reflectance-map model of light reflection. Appropriate combinations of ground cover and sun position can be used to give the user the best possible feel for the mountains and hills that constitute the terrain.

Interestingly, however, shaded relief maps need not conform to what might actually be observed. Horn has made images that correspond to terrain illuminated by three suns, one blue, one red, and one green. Such images give special insight into terrain properties at a glance. Slopes with exposure to the south, for example, are readily identified because of their red hue from the red, southern sun.

The thrust of Horn's work, however, is to make images that match photographs as closely as possible with a view toward registering real aerial photographs with terrain models. Such matching is a vital first step toward improving the use of satellite images.

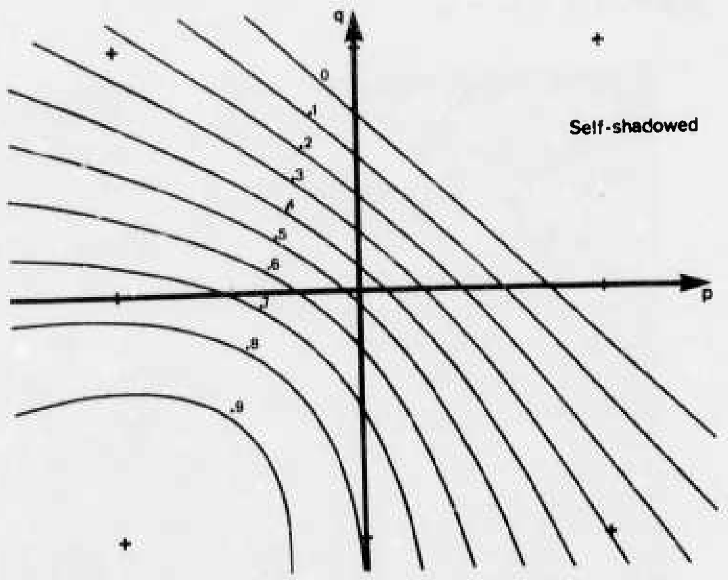


Figure I: A reflectance map. Lines are loci of constant reflectance, showing the combinations of partial derivatives that reflect light equally. This map assumes illumination from the lower left.

Albedo Images and Change-detection Images

After a real aerial photograph is registered with a synthetic one produced from a terrain model, some areas will refuse to match well because the actual ground cover is not the one assumed in generating the synthetic image. Horn defines an *albedo image* to be an image in which each point's intensity is the ratio of the intensity in the real image to the intensity in the synthetic image. In addition to use in classification, it seems likely that albedo images will be useful in change detection. It would be nice if change could be detected by subtracting one image from another. Unfortunately, the changes in sun position from hour to hour and from day to day make this impossible by swamping changes caused by changes in the ground cover. Instead, Horn proposes to divide earlier and later real image intensities by the intensities predicted by the terrain model to give two registered albedo images. Then, one albedo image is subtracted from the other, producing change that will correspond to ground-cover differences occurring between the earlier and later recording times.

For human use, the two albedo images can be printed in different colors and superimposed. The human analyst's eye is instantly drawn to places where changes have taken place because their hue will differ from the surrounding area.

Marr's View of Vision Theory Stresses Three Representations

Marr has championed the idea that vision research must follow these steps:

* First, a competence to be understood is precisely described. Often this means understanding the limits of the various modules of the human vision system. Knowing the strength of the various modules in an existing, clearly good system, helps us to know what competence is needed in the modules of the computer-based systems of the future.

* *Second, representations are selected or invented that facilitate explicit description of the target processing products.*

* Third, the competence and the representations are combined into a well-defined computation problem to be solved.

* Fourth, algorithms are devised that perform the desired computation.

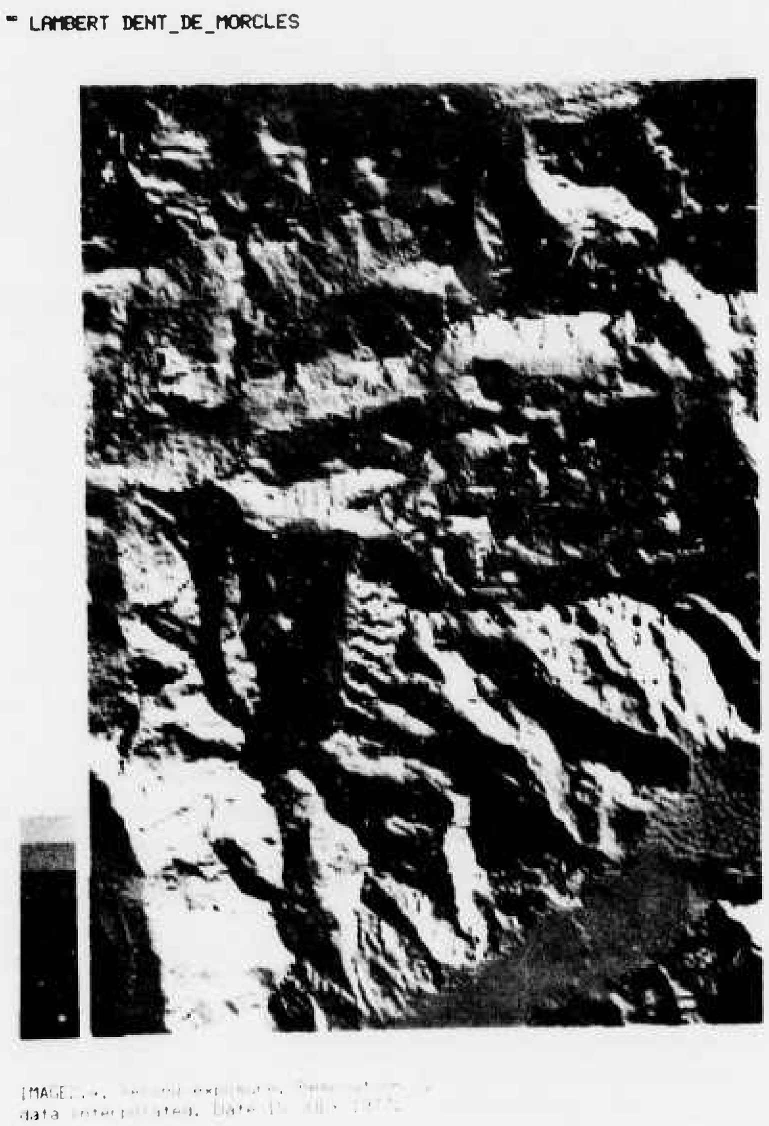


Figure 2: A synthetic images. THis one, a piec[illegible] Switzerland, was made using the reflectance map of figure 1.

* And fifth, results are validated by computer implementation.

At the highest level, observation of competences and definition of representations have led Marr to think in terms of three levels of representation. The *primal sketch* makes information about intensity changes explicit. The *2 1/2 D* sketch makes information about surface orientation explicit. And the the *3-D model* makes information about object shape explicit.

Zero-Crossings and the Primal Sketch

The raw primal sketch is constructed from the image, producing primitive description of the intensity changes in terms of edges, bars, blobs, and terminations, which are each characterized by positior, o[illegible]ntation, contrast, and size.

Computing th[illegible] aw primal sketch falls naturally into two parts: (i) since intensity changes occur in natural images over a wide range of scales, we first detect and represent the intensity changes at a set of different scales; and (ii) the descriptions that arise from these independent channels are then combined into a single primal sketch of the image.

Marr and Hildreth have shown that provided some weak conditions are satisfied, intensity changes at a particular scale in an image I(x,y) are best detected by locating the zero-crossings of $D^2G(x,y)*I(x,y)$, where G(x,y) is the two dimensional Gaussian distribution, and D^2 is the Laplacian. The operator D^2G uniquely satisfies certain critical properties of localization in space and frequency. The smallest operator usable in practice has a diameter of 9 picture elements in the central, positive region, with an overall support of roughly 1000 pixels. Interestingly, this is roughly the size of the smallest channel found in early human vision. The zero-crossings are then represented by a set of oriented primitives called zero-crossing segments, each describing a piece of the contour whose intensity slope (rate at which the convolution changes across the segment), and local orientation is roughly uniform. Small, closed contours are represented as blobs, also with an associated orientation, average intensity slope, and size defined by their extent along a major and minor axis.

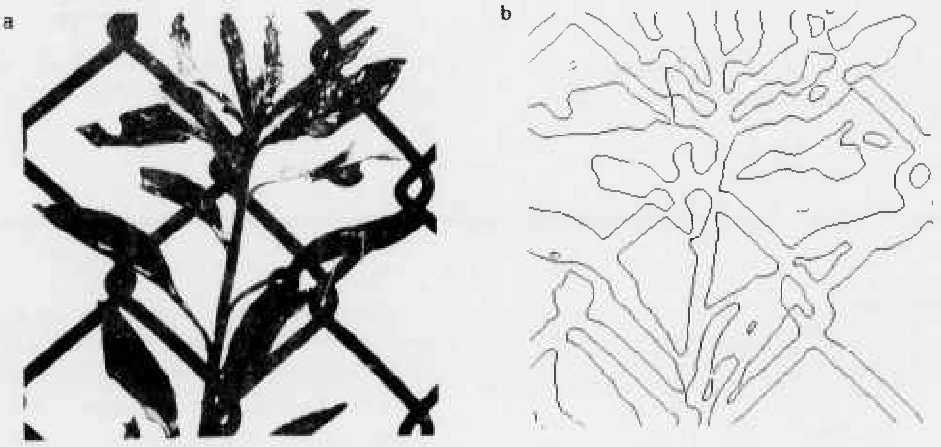


Figure 3: Zero-crossings in the output of the convolution of the above image with a D^2G filter.

Some intensity changes will give rise to zero-crossings over a range of adjacent scales, while others may be detected only at a single scale. In combining information from the separate channels, we take advantage of the observation that intensity changes in an image arise from surface discontinuities, or from reflectance or illumination boundaries, which all have the property that they are spatially localized. This observation led to the spatial coincidence assumption, which states that if similar information concerning the presence of an intensity change is found across a set of adjacent channels, they most likely describe the same physical intensity change, so their descriptions may be integrated into a single description of an edge. Information in one channel which does not coincide with that from adjacent channels is assumed to arise from a physical phenomenon which can only be measured at that one scale, so it gives rise to an independent descriptive element. The final raw primal sketch contains a binary map specifying the position of original zero-crossing contours, together with the symbolic description of the intensity changes, obtained from the separate channels. Figures 3, 4, and 5 illustrate these components of the raw primal sketch. Figure 3b shows the map of zero-crossing contours for the image in Figure 3a, while figures 4 and 5 show symbolic representations of some of the descriptors attached to the locations marked in figure 3b. Figure 4 illustrates the blobs detected in the image, and figure 5 the local orientations assigned to edge segments. These diagrams show only the spatial information contained in the descriptors. Typical examples of the full descriptors are:

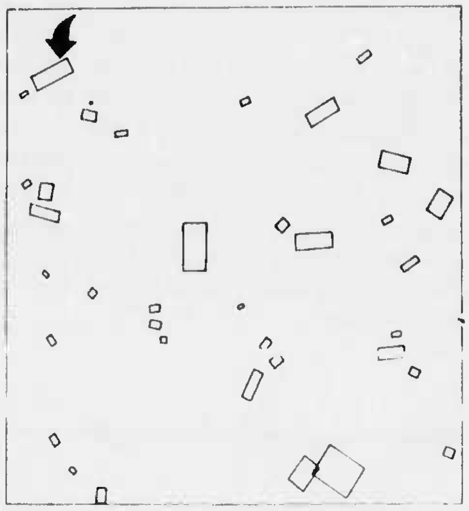


Figure 4: Symbolic representation of the blobs detected in the image. The arrow marks the blob whose full descriptor appears in the text.

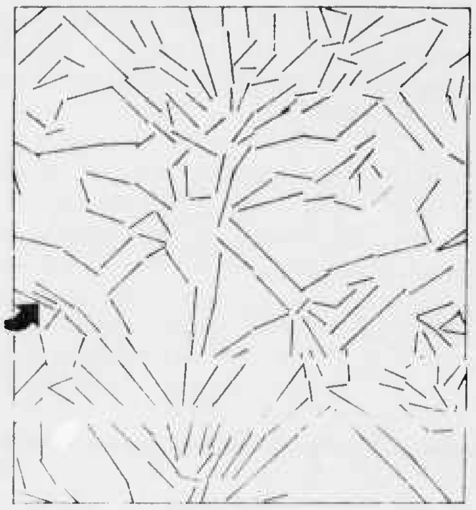


Figure 5: Symbolic representation of local orientations assigned to edge segments detected in the image. The arrow marks the blob whose full descriptor appears in the text.

```
(BLOB (POSITION 146 21)
      (ORIENTATION 105)
      (CONTRAST 76)
      (LENGTH 16)
      (WIDTH 6))

(EDGE (POSITION 104 23)
      (ORIENTATION 120)
      (CONTRAST -25)
      (LENGTH 25)
      (WIDTH 4))
```

The descriptors to which these correspond are marked with arrows.

The 2 1/2 D Sketch

The 2 1/2 D sketch makes information about surface orientation explicit. Figure 6 illustrates what is in the 2 12/ D sketch.

Body-centered 3 D Models

The key to shape recognition is to produce as consistent a description as possible of shape from the local surface information available in the 2 1/2 D sketch. The description should not, for example, depend on the viewer's vantage point. Marr and Nishihara have stated the problem formally in terms of three criteria, *accessibility*, *scope* and *uniqueness*, and *sensitivity* and *stability*. From this they determined that to be suitable for recognition a shape representation should be (1) based on the arrangement of volumetric features such as centers of mass and axes of elongation or symmetry, (2) that these arrangements should be specified in an object-centered coordinate frame (as opposed to a viewer-centered one like that of the 2 1/2 D sketch), and (3) the description should be modular with each module specifying the relative arrangement of a small number of related features which could stand alone as a shape description.

One example of such a representation is that of Nishihara which extends the generalized cylinder representation invented by Binford at Stanford. Figure 7 illustrates the representation.

Nishihara's thesis deals with the problem of computing such a description from the 2 1/2 D sketch. The work includes a consideration of a technique based on identifying chains of local ridge points at a given resolution and over a range fixed by the resolution. The results are not complete but early indications are promising and further work is in progress.

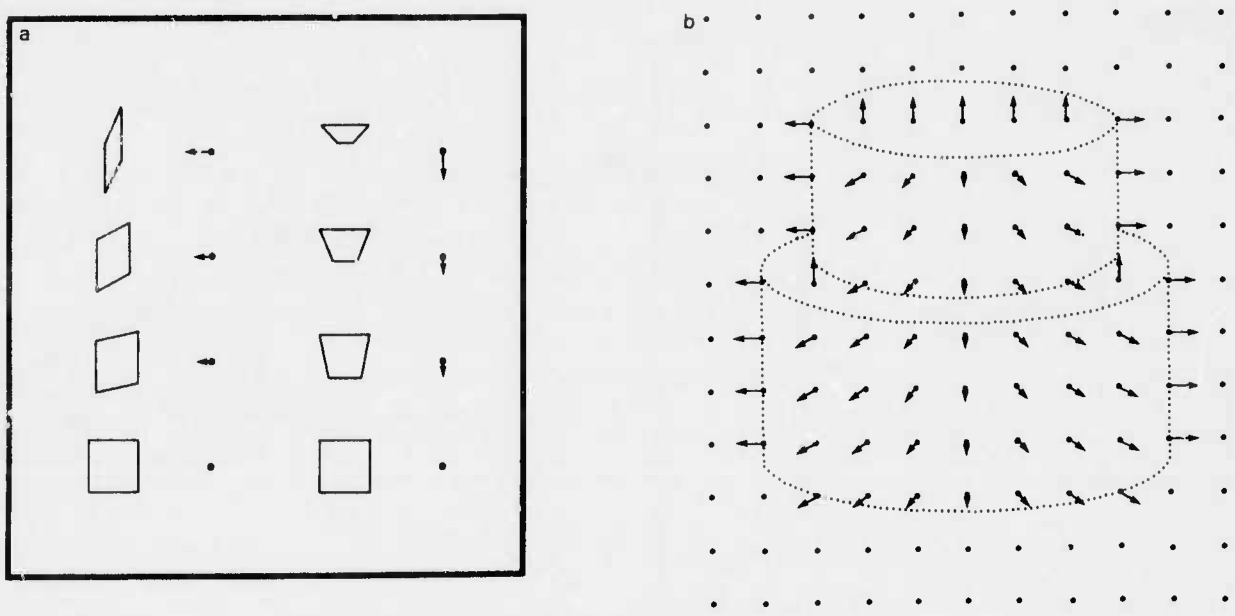


Figure 6: The 2 1/2 D sketch represents depth, contours of surface discontinuity, and the orientation of visible surfaces. The orientation of the needles is determined by the projection of the surface normal on the image plane, and the length of the needles represents the dip out of that plane (part *a*). A typical 2 1/2 D sketch appears in *b*, although depth information is not represented in the figure.

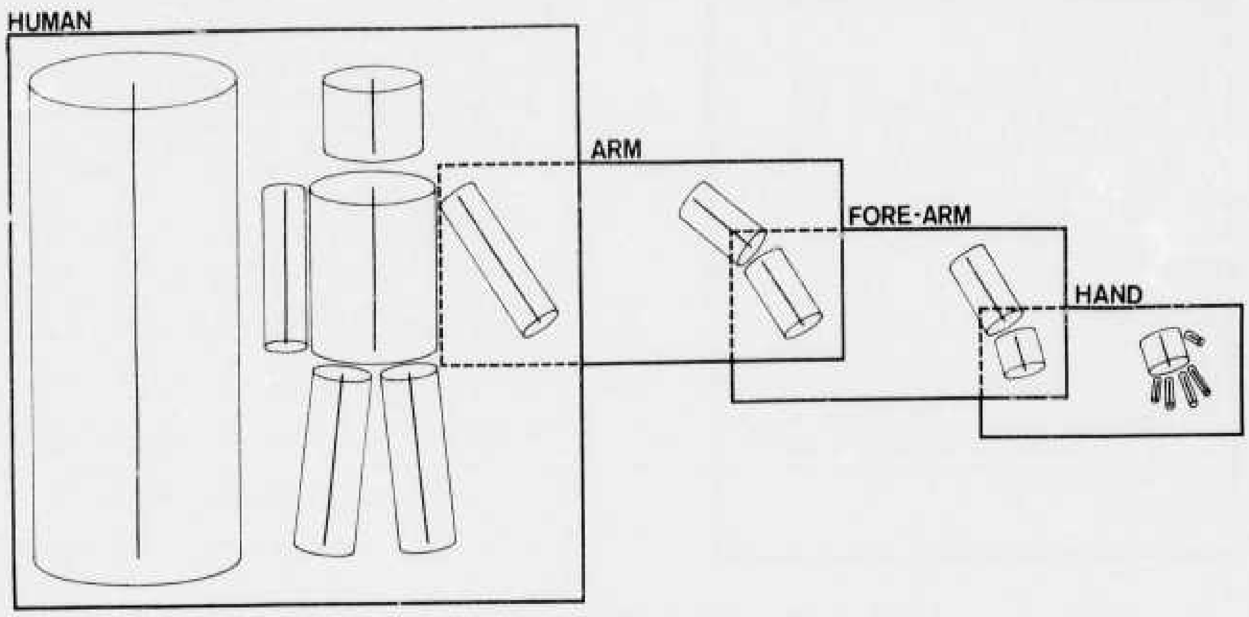


Figure 7: This diagram illustrates the organization of shape information in a 3-D model description. Each box corresponds to a 3-D model; with its model axis on the left side of the box and the arrangement of its component axes are shown on the right side. In addition some component axes have 3-D models associated with them and this is indicated by the way the boxes overlap. The relative arrangement of each model's component axes, however, is shown improperly since it should be in an object-centered system rather than the viewer-centered projection used here. This example shows a coarse overall description of a human shape along with an elaboration of one of its components (the arm). The important characteristics of this type of organization are: each 3-D model is a self-contained unit of shape information and has a limited complexity; information appears in shape contexts appropriate for recognition; and the representation can be used flexibly (components can be elaborated according to the needs of the moment or the time available, and a 3-D model description of a component is easily added to a description of the whole shape).

REFERENCES

For an extended discussion of MIT work on Image Understanding, see Volume 2 of *Artificial Intelligence: an MIT Perspective*, edited by Patrick H. Winston and Richard H. Brown, MIT Press, Cambridge, Massachusetts, 1979.

Berthold K. P. Horn and Brett L. Bachman, "Using Synthetic Images to Register Real Images with Surface Models," AIM-437, The Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1977. Also in *CACM* November, 1978.

Berthold K. P. Horn and Robert W. Sjoberg, "Calculating the Reflectance Map," AIM-495, The Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1978. Also to appear in *Applied Optics*, June, 1979.

B. F. Logan Jr., "Information in the Zero-crossings of Bandpass Signals," *Bell System Technical Journal*, 56, 487-510, 1977.

David Marr, "Representing Visual Information," AIM-415, The Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1977.

David Marr and Ellen Hildreth, "Theory of Edge Detection," submitted for publication.

David Marr and Keith Nishihara, "Representation and Recognition of the Spatial Organization of Three-dimensional Shapes," *Phil. Trans. Roy. Soc. B. 275.*

David Marr, T. Poggio, and S. Ullman, "Bandpass Channels, Zero-crossings, and Early Visual Information Processing," AIM-491, The Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1978.

Representation and Communication of Image-Related Information

Kenneth R. Sloan, Jr.

Department of Computer Science
University of Rochester
Rochester, NY 14627

ABSTRACT

The display, transmission, analysis and storage of images present many problems having to do with both the images themselves and other, image-related information. Representation of the image-related information is especially difficult in a multiple-machine environment.

At Rochester, we have been developing representations and communication strategies for images. The "Name-Type-Value Slot" is emerging as a powerful, general structure. It is proving to be especially useful for the representation of information about the image, the description of the image representation, and possibly the encoding of the image itself.

1. Introduction

Image processing and image understanding (the application of artificial intelligence techniques to image processing) require the storage of, communication of, and access to very large amounts of image data.

Images are commonly represented as a two-dimensional array of picture elements (pixels). Each pixel represents a small area in the original image. The value of a pixel is the "color" of the original image over that small area. Such images are known as "raster" images. The color measure may be binary, grey scale, or a multi-spectral vector. Other ("non-visual") information such as texture measures, population, or land use catagories also may be represented as images.

The primary use of raster images is, of course, display. Towards that end, the images may be manipulated to select sub-images, enhance the features of interest, change the range or distribution of "color" values, etc. They may be analyzed by computer programs or by humans using interactive aides. Such analysis may involve the application of segmentation techniques, texture measures, or "queries" to the image about the location of specific features. In all of these operations (display, enhancement, analysis), derived images are produced and stored for later processing. In addition, these operations require as input, or produce as output, symbolic information which is about the image, but is not the image itself.

This range of uses and the multitude of images produced in a significant image processing effort results in a wide range of image management problems. To store images, one must have a representation that allows the image to be reconstructed and accessed effectively. There are many problems in sharing images, especially those with different characteristics, among programs. These problems multiply when several different machines are being used. Problems involving maintaining relationships amoung images, such as derivation histories, and the useful catagorizations of images are non-trivial. Finally, the storage and maintenance of derived auxiliary information, as from scene analysis, present more problems.

In this paper, we discuss the use of a single, uniform data element--the "Name-Type-Value Slot." The NTV slot stems most recently from the work on PLITS [Feldman, 1979]. We will consider its use in the context of the representation of images, the description of such image representations, and the representation of information which relates the image to the rest of the world.

2. Classes of Image Information

The information associated with an image (or a collection of images) can be classified into several broad catagories. Each class of information has a different pattern of use and generally requires a different storage technique.

First, there is the image itself. Essentially, this is a two-dimensional array of (possibly vector, usually integer) values. Storage of this data is relatively straightforward, and there are the usual problems in array storage and description.

This leads us to the second class of image related data: Format Descriptors. We need to know the size of each pixel, the dimensions of the image array, and how the pixels are packed. This is the basic information needed in order to locate the pixel at location [x,y] in the image.

A third class of image related information is that which is derived directly from the image itself. Some of this derived information is very expensive to compute but requires very little storage (at least relative to the image). Statistics computed over the pixel values (min, max, mean, standard deviation), or histograms of these values fall into this catagory. Information about the results of a feature detector (edge, blob, circle, etc.) may also be of this type.

Sometimes the derived information takes the form of another image. Image operations such as windowing or sampling reduce the size of the image array. Intensity transformations change the values of pixels, and sometimes the number of bits needed to represent the pixels. The results of a region grower or edge finder may be represented as an image with (hopefully) the same form as the original image but with "colors" chosen from another space. A Fourier transform produces an "image." All of these derived images are related in some way with the original image, and these image-image relationships form yet another class of image-related information.

Finally, there are image-world relationships. These may be known a priori, or may be the result of an analysis of the image. For example, the mapping of pixels in an aerial image to locations on the ground may be the result of one stage of image analysis and the starting point of another.

This classification of image-related information is based on function and intent. Some of the classes are easy to characterize in terms of the amount of storage required, but others are quite variable. All pose problems in storage, access and maintenance of needed relationships with other information.

3. Name-Type-Value (NTV) Slots

Name-Type-Value slots are self-describing units of information with a NAME, used for accessing the information, a TYPE, describing the representation, and a VALUE, the actual value of the slot.

Part of its heritage comes from the LEAP concepts developed for the language SAIL [Feldman, 1969]. In addition, it shares several concepts with the PLITS message-based system designed at Rochester, including the actual internal format for the NTV slots [Low, 1978].

The underlying meaning of an NTV slot is that it describes a relation. For example, the NTV slot <"Date Photographed"-STRING-"27OCT72"> might be used to encode the fact that the image involved was photographed on 27 October 1972.

In general, NTV slots are accessed by NAME. These NAMEs are uninterpreted strings which are chosen for their mnemonic (to the user) value.

The TYPE describes the representation of the VALUE. Together, the TYPE and VALUE provide the information specified by the NAME. Two major design decisions characterize the TYPEs used in NTV slots. First, there are a small number of basic types, including STRING, BOOLEAN, INTEGER, REAL, and ARRAYS of these. Second, the representation of each of these is of variable length and relatively machine independent. The set of available, basic types reflects our current desire to provide a moderate amount of descriptive power, without overly complicating the system at the level of the individual slot. On the other hand, the reality of many machines, with many different representations for even such basic data types, motivates the decision to allow a great deal of flexibility in the lengths of data items.

Several examples of NTV slot specifications are shown in figure 1:

```
<"TITLE"-STRING-"Reference Picture">
<"Date Photographed"-STRING-"27OCT72">
<"LENS SER NO."-INTEGER-799648>
<"Date Digitized"-STRING-"15JUL79">
<"Pixel Width in Meters"-REAL-2.50>
<"Pixel Height in Meters"-REAL-2.50>
<"Major Interchanges"-ARRAY STRING-
  ("Left Center",
   "Left Bottom"
  )
 >
<"River"-STRING-"Right Top to Left Bottom">
```

Figure 1: NAME-VALUE Slot Specifications

4. Current and Future Uses

The current formulation of NTV slots, and our understanding of their usefulness in image understanding work, is the result of extensive experimentation with earlier incarnations of similar ideas. The next iteration of this process will involve basic changes in the ways that we deal with images and the information associated with them.

4.1 RIFF

One major example is the incorporation of NTV slots into our local Raster Image File Format (RIFF) [Selfridge and Sloan, 1979]. As implemented in RIFF, NTV slots are simple and general. Any NTV slot, either constructed in a program or defined by a user with an interactive program, can be stored in image file headers with a simple command. Their access by NAME means that their order in the header is irrelevant. In addition, their variable VALUE size allows information to be stored regardless of length. This contrasts with other image information formats ([Hayes, 1975; Tamura, 1977]). The information represented by NTV slots can be user-defined, and can include annotations, image statistics and pointers to other files representing image-image relationships. Their direct association with image files is convenient where one wishes to move images between components of a distributed system.

NTV slots, as used in RIFF, are not suitable for all classes of information. Complex relationships, especially between images and other kinds of information, such as symbolic ones, are probably best done in other representations such as image data-bases [Mckeown, 1976]. However, NTV slots do allow possible links to such alternate representations.

We are currently using RIFF-NTV slots in several ways. General annotations include a descriptive TITLE slot, far more useful for identification of image files than their six character file name. Images which are windows of large files hold two slots. A WINDOW PARAMS slot holds the window coordinates in the larger image, and an ORIGINAL slot holds the name of the original image file. The program which acquires an image from our drum scanner and builds an image file uses RIFF-NTV slots to note the time and date of the digitizing session, the interpretation of the samples (e.g. Red, Green, Blue color separations vs. Grey Scale), and the TITLE.

Another use of RIFF-NTV slots is to hold image statistics. These are stored with memo functions, functions which automatically annotate image headers with computed statistics. These include the average, minimum, maximum, and mean pixel values, and the standard deviation. These are very costly to re-compute, so storing them in the header using pre-defined slot NAMEs greatly increases their usefulness.

Using slots as file pointers can be useful in many situations. They can associate derived images, as in the window example above. Other examples might include thresholded or convolved images. In addition, sets of images can be constructed by using file pointers to form linked lists. For example, all images derived from the reference pictures can be related in this way, allowing people and programs to access them as a unit.

4.2 Image Data Bases

We have also used a more complex scheme, using file pointers to make a structured data base of images, related in various ways by different kinds of pointers. An interactive "file fetcher" can access this structure at certain "roots" to locate or create new sets of images and answer specific queries. Such a structure may only be useful in a limited domain, and is limited by the cost of chasing pointers through many files, but it offers easy entry to image data base experimentation. This is especially important to us because of the multitude of machines, each with a different file system, involved in our work. The ability to have a small kernel of symbolic information permanently attached to the image file itself has proven to be very useful.

On the other hand, the NTV slot, as an isolated unit, appears to be the right form for the communication of image-related information stored in a (central or distributed) large scale general data base which has been extended to deal with image data.

4.3 Image Transmission Protocols

Recent work at Rochester has included the investigation of image transmission schemes which allow for the effective use of a distributed set of vastly different machines, connected by relatively slow communications lines [Sloan and Tanimoto, 1979]. This work is now ready to be combined with the NTV and PLITS ideas, with the goal of allowing a distributed community of processes to negotiate the amount and form of image data to be transmitted among themselves. This will be especially important in the effort to extend large scale data bases to include images as a basic data type.

5. Conclusions

Image management, including managing image files and additional information about images, is a hard problem. There are many aspects to the problem, including relating information to an image, images to each other, and images to symbolic representations of what the image contains.

Likewise, there are many levels of solutions. RIFF is a compromise between fixed information and full generality. It provides the portability of a fixed format for the image and its format descriptors, along with a simple, general and flexible method of storing image-related information of several kinds in image headers.

The general format of NTV slots, coupled with the relative independence of each slot, invites the development of more complex schemes. These include the integration of large scale data bases into a truly distributed image understanding system, and the extension of such data bases to include the image as a basic data type.

REFERENCES:

Feldman, J.A., "High Level Programming for Distriuted Computing," CACM 22, 6, pp. 353-368, June 1979.

Feldman, J.A. and P.D. Rovner, "An ALGOL-Based Associative Language," CACM 12, 8, pp. 439-449, August 1969.

Hayes, K.C. Jr., "XAP User's Manual," TR-348, University of Maryland, revised 1975.

Low, J., "PLITS Memo 4: Internal Format," Internal Memo, Department of Computer Science, University of Rochester, Rochester, NY, 1978.

Selfridge, P. and K.R. Sloan, Jr., "Raster Image File Format (RIFF): An Approach to Problems in Image Management," Proceedings PRIP 79, IEEE-CH1428-2C, pp. 540-544, 1979.

Sloan, Jr., K.R. and S.L. Tanimoto, "Progressive Refinement of Raster Images," TR39, Department of Computer Science, University of Rochester, November 1978.

Sproull, R.F. and P. Baudelaire, "Proposed 'Array of Intensity' Samples Format," Xerox PARC Internal Document, May 1976.

Tamura, H. and M. Shunji, "A Data Management System for Manipulating Large Images," in Proceedings of the Workshop on Picture Data Description and Management, IEEE-77CH1187-4C, April 1977.

SPATIAL REPRESENTATION

Thomas O. Binford

Artificial Intelligence Laboratory, Computer Science Department
Stanford University, Stanford, California 94305

Abstract

An aerial photograph of the Fort Belvoir area is used to illustrate representation levels of ACRONYM. Image observations and organization of image features are described. Relations between objects in context provide strong conditions for searching and verification.

ACRONYM contains a variety of spatial representations and it will integrate others when they are implemented. Figure 1 shows a diagram of levels in ACRONYM. These levels are: 1. The context level contains functional groups of objects, e.g. airport, parking lot, and residential area. 2. Object level includes specific objects and generic objects, i.e. object classes. 3. Volume level is a three space level which contains elements such as cylinders. 4. Surface level is also a three space level the elements of which include planes. 5. The image structure level is a planar level the elements of which are two dimensional structures abstracted from images. 6. The image is a projection of a scene (two dimensional) recorded by any imaging sensor. Representations at each level are shown in Figure 2.

Context Level

The context level is a graph of functional relations among objects. It is not clearly distinguished from the object level, however there are two senses which prompt this distinction in name. The first sense is that everyday physical objects like houses, aircraft, and cars are more tightly defined than abstractions such as a residential area or airfield. The second sense is that objects have relationships determined by a higher function. Internally, the context level is represented in ACRONYM within the Object Graph, as a graph of relations among objects.

In the standard aerial photo of Fort Belvoir, consider the context level. This is an area of natural terrain, superimposed on which are several types of cultural clusters, linear cultural features, and linear natural features. This is a low development area. The clusters include: a residential area at left edge, middle; another at center, bottom edge; numerous clearings without buildings and with various patterns; clusters of larger buildings (circular, cross shape, elongated) all of which are low. Cultural linear features are roads and their interconnection pattern to other roads and connectivity to building clusters. The river is an obvious natural linear feature. The natural terrain is wooded.

Roads are described as a transportation network for cars and trucks. Types of vehicles and interconnections and terminations are specified. Cloverleafs and crossings, bridges over water, typical grade limits and cut and fill are included. Road width is related to traffic volume. Building clusters are described by their connection pattern: the residential cluster at the left edge, middle is a branching linear connection pattern. Typical cities have a grid pattern. Houses line roads at regular intervals. With residential buildings are playing fields, power lines, parks, lawns, and gardens. With individual houses are driveways; with multiple dwellings are parking lots. The number of cars or spaces for cars is a measure of people. With farm buildings are silos and fields. With large bodies of water are found boats and piers. Tanks come in clusters; there are relatively tall tanks on the left bank, lower tanks on the right bank (white) and a cluster of tanks in a cleared area away from the river. Streamlines near the river run roughly orthogonal to it. Streamlines show the direction of surface slope. Water is a local minimum height. Straight lines and regular boundaries suggest cultural objects.

Object Level

The Object Level is represented by a graph of relations between objects. Objects are represented by examples; object classes are defined by function. Objects have volume representations and may be defined by their volumes. At this stage in ACRONYM's development, objects are represented primarily by their volumes. Object classes may be defined secondarily by volume descripters with more general quantification than that for specific objects.

Water in ponds and lakes is level and often very smooth. It has strong specular reflection, i.e. large variation in reflectivity with angle. Note that water is black near the center of the picture and much lighter near the edges. Roads are made of concrete, asphalt, dirt, or grass-covered. Most road and streets are concrete and bright in this example. At bottom near the center is a cluster of houses with dark streets and trees along the streets. One area has sidewalks which appear as bright lines.

Houses are a typical size. This grouping may be more uniform than typical scenes, but houses tend to be clustered in similar sizes and lot sizes. Buildings themselves are described by their function: to enclose space as a shelter for some set of people or objects. Sizes depend on the set of people and/or objects. Structures are relatively modular, composed of units related to these functions. A tall building in an area of low concentration may house tall objects. There are few tall buildings in this example (a silo). Circular buildings are unusual in this example and typically. Houses have vertical walls usually, for structural reasons. Usually walls are planar at right angles. Roofs are sloping or flat depending on snow load. Few buildings have shell structures. Trees are vertical for the same reason, gravity. Trees are discrete objects with typical size and density dependent on species. Rails carry heavy objects, hence railroad bridges are reinforced.

Volume Level

Volumes are represented as part/whole graphs, directed graphs whose nodes are generalized cylinders and whose arcs are attachment relations. Spheres are included. Many other specific prototypes are special cases of generalized cylinders. Other volume representations include three-dimensional Blum transforms, three-dimensional incidence matrices (e.g. unit cubes which are filled or empty), density distributions represented by orthogonal polynomial expansions, and polyhedra. These are not used in ACRONYM. Numerous questions arise about transformations from one representation to another.

We call this latter group non-symbolic representations; they either tabulate data or approximate data without segmentation in analogy with curve fitting and surface fitting. Symbolic representations adjoin names, relations, and segmentation to non-symbolic representations. Names, relations and segmentations are all related to task and scene content and reflect constraints from models and external knowledge. These constraints permit more accurate determination of task parameters than uninterpreted data.

Surface Level

Symbolic surface representations are built with ribbon elements. Non-symbolic surface representations are also included at the surface level. Surfaces are embedded in three-dimensional space. Ribbons are generalized cylinders restricted to surfaces. Ribbons are closely connected to surface splines. Surface splines are piecewise surfaces made of elements which are usually polynomial with specified continuity conditions at their seams. For example, a spline may have continuous tangent plane and continuous curvatures at its seams. Typically splines are non-symbolic in the sense above; their seams are assumed arbitrary independent of the data (e.g. on a fixed grid) or supplied by the user. However, we are interested in non-uniform continuity conditions along seams which are determined from the data. That is, we are interested in splines with discontinuities in position and tangent plane over some seams and with continuous tangent plane and curvature along other seams. These are like cardboard and rubber cutouts superimposed to cover the scene. Little work is done in numerical analysis on free knots or seams. This segmentation problem is inherent in scene analysis. Well-founded solutions or ad hoc solutions contribute to symbolic surface description.

Stereo and ranging devices produce tables in the form of surface maps of range vs azimuth and elevation angles or surface maps of elevation vs x and y camera coordinates. Stereo also may produce segmented symbolic boundary information superimposed on surface maps; boundary information includes high accuracy estimates of depth and slope discontinuities. Motion also produces surface information given an interpretation of corresponding surface features in terms of rigid body motion or articulated rigid body motion. Object motion is equivalent to observer motion if the background is ignored.

Prominent surface elements in this image are roads, which are regular, constant width, mostly bright ribbons. In some places they involve cuts or fills which show up as bright irregular areas. Excavation shows up as bright areas. Fields are approximately rectangular with linear texture. Terrain is represented as ribbon surfaces particularly at ridge lines or stream lines. Streams show up as irregular dark linear features. Wooded areas have dot patterns of individual trees. Tanks show appear as circular areas with shadows. The water surface and road surface are good places to find shadows, e.g. bridges and trees.

Image Structure Level

Image structures are built up from images. Images are embedded in two dimensions, usually the plane or the unit sphere. The lowest level of representation involves edge elements called edgels which are local discontinuities of the intensity surface, defined over small neighborhoods. Extended boundaries of the intensity surface are constructed from edgels. Intersections of boundaries at vertices and closed region boundaries are formed from extended boundaries. Ribbons are useful descriptors for both closed regions and open boundary configurations. They provide a useful sense of locality which is stronger than topological connectedness. Regions and open boundaries are grouped into texture regions. These operations define a hierarchy of levels. Vertices of curves and terminations of curves are special points which may be grouped into curves. Regions or segments of region boundaries may be grouped into curves, or grouped into regions. Texture regions may be grouped in the same way into curves or into texture super-regions.

Linear features appear strongly. There are both regular linear features and irregular ones. At the level of connected curves and intersecting curves, most of the roads and streets are delineated. The road through the residential area in left, middle part of the picture is quite regular but broken, a dotted line boundary. Small linear features of large buildings also appear. Even smaller features like houses prove on closer examination to have straight line edges. However, curves from roads, parking lots and large buildings provide most of the low level context. Homogeneous regions which we recognize as clearings, water, and roads also delineate areas of interest. Regular features are of special interest for cultural features.

Linear features and dot features are grouped into structures and textures. For example, clusters of houses are homogeneous in size, orientation, and relative vectors. They are structured along large linear features. Larger buildings have similar structure.

Although buildings are regular and can be distinguished by form and textural relations, in this image a lot of work is necessary to cover the whole image at a resolution sufficient to distinguish potential buildings from trees to take a closer look. Consequently, depending on the task, buildings are likely to be isolated by looking along roads or in clearings. A typical task is to find tall buildings. The low level context of roads is enough to make a linear search restricted to roads.

Much interpretation in that image is made possible by shadows. These are useful where shadow edges are extended and approximately parallel to the edge casting the shadow, as in the bridges. Also, where shadow regions are adjacent to shadow casting structures, as with houses, tanks, and trees there is little trouble in associating shadows with objects.

There is a dark irregular patch with a sort of triangle in the center. It appears to be a marking. What is it?

Tasks are represented as functions to restrict quantification of object descriptors. Count all oil tanks with volume v>v0 becomes:
Cardinality {x|xoil_tanksvolume(x)>v0}.
Measure total volume of all oil tanks becomes:
Sum{volume(x)←x|xoil_tank}
in an informal notation. Sequences contain an indexing parameter which may be an integer, time, distance, angle, or other; they relate to a physical parameter, typically time or distance, by means of a generating function which may involve other parameters such as velocity. Sequences may be synchronized at points with other sequences.

Information comes in at all levels. The information must be integrated by forming correspondences within levels and by mapping between levels. Identification/interpretation is at the volume level, based on correspondences at all levels. We believe that form=function arguments connect the volume level fairly directly to object interpretations. Most signal information enters at image level and surface level. Signal information includes reflected intensity for a variety of spectral components, range, and Doppler information. Low level symbol descriptors obtained from signals include edges, ribbons, and stereo depth maps, which provide information at the image level and surface level.

Generalized cylinders correspond to stacking volume elements like slices of bread. Ribbon surfaces correspond to stacking surface elements. Generalized cylinders are represented by a cross section translated along a space curve called the spine. The cross section is transformed by a sweeping rule and there are terminations at either end of the generalized cylinder. Generalized cylinders are determined by the principle of generalized translational invariance.

We aim to represent elements by mathematical entities and to relate these entities by maps within levels and between levels. Several of the levels correspond to entities of three dimensions (volumes), two dimensions (surfaces), one dimension (curves), and zero dimensions (points) embedded in spaces of three dimensions, two dimensions and one dimension. We have maps which decrease dimension (projections) and maps which raise dimension (sweeping operations).

Our work has been based on the following paradigm. Descriptions are made of geometrical entities formed by two processes: a. grouping by a few geometric relational operations which are more or less independent of the entity and which are common to all levels; these grouping operations correspond to neighborhoods of approximately uniform shape, elongated narrow neigborhoods in all directions corresponding to longitudinal projection, and transverse projection [Nevatia and Binford]; b. discrimination by tight constraints which are specific to the geometrical entity.

In the ACRONYM diagram we show models at each level of the hierarchy. There are models with various degrees of generality at each level. In the most specific cases we may predict intensities at the pixel level of the image. In a general case we use models for edges in a small neighborhood. That is, even in a general case, we have local models for what we expect to see, even at the image level. As mentioned above, at other levels models include: extended curves, segmented curves, planar ribbons, ribbon surfaces, and generalized cylinders. Because of computational limits and information limits we have limits on perception. Computational limits imply that we compute only the simplest few of the enormously many possible functions on an image. Information limits imply that we can consider only relatively simple underlying object interpretations for observations. These models represent a commitment or preconception to perceive what we can within those limits. Observations are represented as instantiations of these models.

Generalized cylinders

Generalized cylinders were initially intended for use in visual interpretation of complex objects as a means for a natural semantics for part/whole segmentation. The idea of a segmentation is not new, but the choice of primitive element determines whether the resulting segmentation into parts is useful. Some requirements of adequate representations were described in [Thomas and Binford]. The design criteria which led to the formulation of generalized cones were:

The representation should be locally generated, that is primitives should be generated from local primitives. I see no way of generalizing an enumeration: cube, sphere, cylinder, ..., of globally specified forms. A visual system which is conceptually adequate must deal with a potentially very large set of objects, including a large set which may never be seen. The part/whole segmentation describes one form of generation, but the primitives which go into the part/whole description must be locally generated.

Parts should be defined by continuity. A surface is not a part in a "natural semantics", since a cube has six surfaces, yet a cube is thought of by most as a single part. If we define parts by surface continuity, then only separate objects are parts, and a man standing on a floor is not separate from the floor. If, on the other hand, we define parts by surface tangent plane continuity, then a cube has six "parts".

Primitive parts should be generated from elements which are disjoint and for which a small, finite set gives a good approximation. In the Blum transform [Blum] which is a covering by a minimal set of maximal interior disks, the elements are overlapping circles, i.e. not disjoint. In the Fourier representation, a few eigenfunctions used to construct forms are an overlapping set. This requirement follows from the intuitive clarity of disjoint elements in description and as an aid in segmentation. The covering by a finite set implies that the elements are volumes.

Generalized cylinders have nothing to do with symmetry or with elongation. They were defined by generalized translational invariance. A coin is a fine generalized cylinder. However, description techniques are better for elongated generalized cylinders. [Nevatia and Binford 77].

Each representation introduces a sense of similarity which is natural in the representation. Generalized cylinders were introduced in order to represent locally generated constructions from fundamental geometrical operations. Generalized translation invariance characterized constructions based on translations [Binford 1971]. Generalized cylinders were to be augmented by spheres which characterize constructions based on rotations. One interpretation of the phrase "natural semantic" interpretation is in terms of these fundamental operations. Another is in terms of mechanical construction operations, which include fabrication operations (milling = translation and turning = rotation, extrusion = translation) and assembly operations (insertion, and screwing).

We first used generalized cylinder representations in vision in interpreting parts of objects [Agin 1972, Agin and Binford 1973], then in interpreting structured objects and recognizing them [Nevatia 1974, Nevatia and Binford 1977]. Several issues were addressed. The first issue was recognizing objects from a very large visual memory. How do we relate an observed description to a subclass of previously constructed descriptions? This was addressed through indexing into a visual memory structured on the basis of an attachment hierarchy. The second issue was obtaining generalized cylinder descriptions from surface information. This was done through several variations of implementation of local translational invariance.

It was recognized at once that the representation offered very compact models of complex objects, suitable for graphics and manufacturing applications. A system for object modeling and graphics in terms of generalized cylinders was built [Miyamoto and Binford 1975]. Generalized cylinders were used in predicting curves in images for Verification Vision [Bolles 1976], and in a planning system for automating screw insertion in mechanical assembly [Taylor 1976].

[Agin 1972] G.J. Agin; *"Representation and Description of Curved Objects"* Stanford AI Lab Memo AIM-173, 1972.

[Agin and Binford 1976] G.J.Agin and T.O.Binford; *"Representation and Descriptionof Curved Objects"*; IEEE Transactions on Computers; Vol C-25, 440, April 1976.

[Binford 1971] T.O.Binford; *"Visual Perception by Computers"*; Invited Paper IEEE Systems Science and Cybernetics; Miami Fla; Dec 1971.

[Bolles 1976] Robert C. Bolles; *"Verification Vision Within a Programmable Assembly System"* Stanford AI Lab Memo AIM-295, CS-591, December 1976.

[Marr 1976], D.Marr; *"Analysis of Occluding Contour"*; MIT AI Memo 372; October 1976.

[Miyamoto and Binford 1975] E.Miyamoto and T.O.Binford; *"Display Generated by a Generalized Cone Representation"*; Conf on Computer Graphics and Image Processing, Anaheim, Cal, May 1975.

[Nevatia 1974] R. Nevatia; *"Structured Descriptions of Complex Curved Objects for Recognition and Visual Memory"*; Stanford AI Lab Memo AIM-250, CS-464, ADA003486, October 1974.

[Nevatia and Binford 1973] R.Nevatia and T.O.Binford; *"Structured Descriptions of Complex Objects"*; Proceedings Int Joint Conf on AI, Stanford 1973.

[Nevatia and Binford 1977] R.Nevatia and T.O.Binford; *"Structured Descriptions of Complex Objects"*; Artificial Intelligence 8, 6, 1977.

[Taylor 1976] R. Taylor; *"Synthesis of Manipulator Control Programs From Task-Level Specifications"* Stanford AI Lab Memo AIM-282, CS-560,, July 1976.

[Thomas and Binford 1974] A.J.Thomas and T. O. Binford; *"Information Processing Analysis of Visual Perception: A Review"*; Stanford AI Lab Memo AIM-227, CS-408, ADA003483, 1974.

Figure 1: ACRONYM diagram

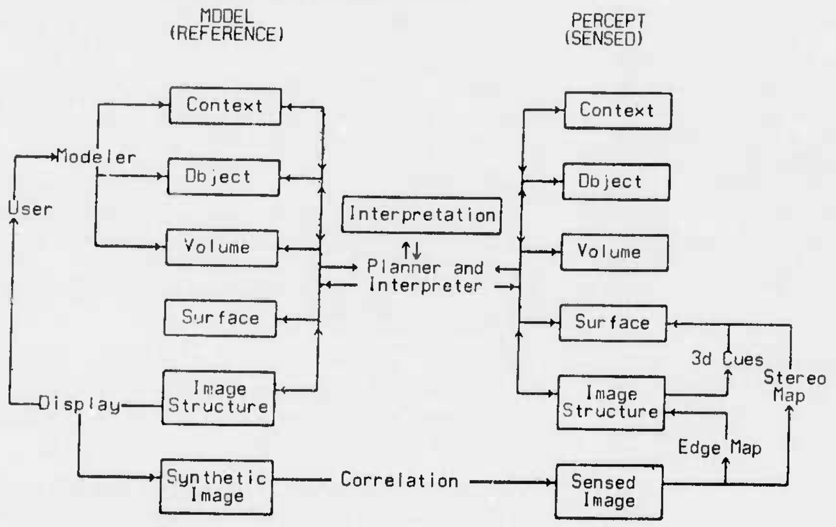


Figure 2: Representation Elements

LEVEL	REPRESENTATION
Context	Objects
Object	Volumes, Surfaces, Functions
Volume	Generalized Cones, Spheres, Surfaces
Surface	Ribbons
Image Structure	Ribbons, Curves, Spots
Image	Pixels

SYMBOLIC REPRESENTATION: USC IU SYSTEM

Ramakant Nevatia
Keith E. Price

Image Processing Institute
University of Southern California
Los Angeles, California 90007

INTRODUCTION

The choice of a symbolic representation is intimately related to the use of this representation for the set of chosen applications. It is believed by many that there exist representations that are useful for a "general" vision system, i.e. one able to perform a variety of tasks, comparable in range to human performance. In this paper, we describe our representations that were primarily designed for the application of image to map correspondence. However, we believe many of the representations, particularly the ones at the lower levels, have general utility, in the sense of being useful for other tasks.

We restrict our presentation to include only those representations that either exist in our current implementations or are planned extensions of these representations. Thus, the described representations do not necessarily reflect our view of how the human visual system works or how a "general" image understanding system might work. The described representations may lack many components desirable for a general system, either because these components are not useful for the current applications, or because the representations are not computable (due to lack of data or of a computing algorithm). For example, we are not using a representation for the range information, such as a 2 1/2-D sketch, because range is only marginally useful for high altitude aerial images and techniques for extracting it from a single image are not adequately developed.

Our presentation is divided in model representations and image descriptions. The model representations are currently input by an interactive system, but they could be derived from a map or an image. The image descriptions are computed automatically, but future implementations may provide for interactive control or editing. The two representations essentially share the same elements. The differences are primarily in that the model representations are ideal cases of computed image descriptions. Also, the image descriptions contain results of computations at intermediate levels that are not necessarily meaningful at the level of model representation.

MODEL REPRESENTATION

We are using a relational graph structure (also called a semantic network) for the basis of our high level image and model representation. Semantic networks ar used by many others in image understanding and artificial intelligence, and all are similar structures except in certain details. The nodes in the network represent elements in the model such as lines, regions, objects of various kinds, and groups of objects. The arcs in the graph correspond to relations between elements. There are several different relations which are used including neighbors, relative positions, nearby, part of, one of. The type of relations which are possible for an element depends on the type of element. The nodes contain the descriptive information about the particular element with the actual information depending on the type. Some of the descriptive feature values may not be known for all model elements. The absence of these descriptor values must be handled by later processing systems and should not adversely affect later results.

Certain metric properties and relations have the same meaning ror all types of nodes. Information on the position of an object in the scene is very important and is expressed in several ways. Locations of the centroid of individual elements in the model are expressed relative to the position of the entire model, which is stored in a node corresponding to the entire scene. Scale information is also kept to convert model units (or image units) to ordinary measurements (meters or miles). Distances between two objects (e.g. 2 km South and 3 km West) may be given as a relation between the two objects but such relations are used only when the spacing between the two objects is very important for the location of one (or both) of them. Such distances maybe computed for other pairs of objects when necessary from their absolute positions. Size and positions can also be given in this form except that the values are multiples of the size of the other object - e.g. twice as large, or one object diameter West. Geometric relations also provide significant descriptive information. Adjacent elements are related by the "neighbor" relation, and objects which are close but not necessarily touching are related by the "nearby" relation. Relative positions are indicated by above/below, to left/right, North/South, etc.

Lines and Linear Features

Linear features are those which can be represented by straight lines of some width, or a sequence of straight lines. These actually correspond to roads, airport runways, narrow rivers, canals, etc. The descriptors which are used include length, width, orientation (direction), position of the center and the end points, and color (bright or dark). The relations include the simple relations of neighbors, nearby, and relative position. Other relations which may involve lines will be covered in the discussion of objects.

Regions

The region features are the simple objects which are not represented by lines. Objects may also be represented by a combination of regions as described in the next section. The descriptors for regions are the same as for lines with some additions: area, perimeter, color (actual intensity values for each input band and other color transformations, computed as the average over the entire region), texture (either structural or pointwise computations), and shape (crude measures which give general descriptions and generalized cones for more precise descriptions when possible). The relations involving regions are the same as those involving lines.

Objects

The region and line elements are sufficient for describing simple objects composed of a single part. We use these simple elements as a basis for descriptions of other, more complex, objects. Objects can be generic or specific, and simple or complex.

Generic objects are used to group similar objects into one class. These are implemented using the relations: "One-of" to denote an object in a class, and "Is-A" to indicate the generic class of an object. Note that these relations are inverses and indicate the same relation. Generic objects are used to ease the description procedure by providing one set of descriptors for all common object classes such as roads, rivers, urban areas, rural areas, etc. These descriptors are used when none are provided for the individual object, so that the descriptor values can be changed for individual objects which have a different appearance. Other relations are generally not meaningful for generic objects.

Simple objects are represented as groups of regions and lines (or other simple objects) using ancestor and descendant ("Part-of") relations to indicate the major object or subparts. The descendants of an object are the subparts of the object and this relationship is labelled part-of. The descriptor values which are associated with objects of this type are used to indicate features of the object as a whole - position, size, orientation. When these features are given the corresponding descriptors of the subparts are taken relative to the value for the entire object.

Complex Objects

Complex objects are represented by their parts and relations between them, as for the entire model. The parts may be simple objects or other complex objects. The simple objects are represented by one of the above described means. As example, an airport would be described by its runways, taxiways, and their geometrical relationships. A runway is a simple object and is represented as a line corresponding to its medial axis (this is a very simple case of the generalized cone representation). The geometrical relations are given by the intersections of the runways and the angles between them.

Example

Figure 2 shows part of the model containing some of the major features and their relations for the image shown in Figure 1 (a complete model would be too large and complex to illustrate in a figure). In Figure 2, the IS-A arc connects specific objects to generic objects, North-of and West-of indicate relative positions, and part-of should be interpreted as the pointed node is a part of the other node. Figure 3 shows the details of the node corresponding to the south portion of the river. Note that some of the descriptive properties are attached to this node, while the others are inferred from the node being a water area having some generic properties.

IMAGE DESCRIPTIONS

The current symbolic representation of images is in the same format as the models described above. This common description method has meant that descriptive features which re used in the model must be computable from the given images. Also this means that there must be precise algorithmic descriptions of the meaning of the relations and features. One major difference is that, initially, the collections of the basic elements are not available. Grouping elements into objects and labeling these objects is performed by various matching procedures and is a major goal of the image understanding system. The labels are implemented as pointers to the model description (i.e., relations between elements in otherwise separate descriptions), or by adding new elements to the image representation to collect the object parts together. The image representation is generated automatically, so the region type elements come from our region based segmentation system and the line type elements come from the linear feature analysis system. The machine segmentation, and hence the resulting descriptions, are likely to be different from the human segmentation used for model generation. The image description also contains descriptors necessary to recover the low level representations of each of the elements, i.e. pointers of the binary masks for regions and to the piecewise linear approximation for line like objects.

The lower level representation of regions in the image is as a binary mask which indicates which points are in the particular region. Only the minimum enclosing rectangle is stored since image offsets are maintained with all image and the data is packed 1 bit per image point. These masks are already small and we require efficient access to both interior points and boundaries so that no attempt has been made to encode these masks in any way.

At the high level, lines are represented by a piecewise-linear approximation and road-like structures by their medial axis. The intermediate descriptions generated in the process of computing these descriptions are also available and possibly useful for feedback from higher level processing. The important intermediate level representations are:

1. Local edge magnitudes and directions, stored as image arrays.

2. "P" and "S" arrays, containing the predecessor and successor information respectively for each edge. As this information is iconic, it may be useful for determining proximity of lines.

3. Lines represented as a list of edge points (rather than by piecewise-linear approximations).

Details of the steps in line finding were presented previously in [1].

APPLICATIONS

We have applied our representation to road finding [1], locating desired structures in aerial images using map information [2], and are designing a system for recognition of complex objects (see [3]). The choice of our representations was guided by these tasks, but we believe them to be useful for other applications as well.

REFERENCES

1. R. Nevatia and K.R. Babu, "Linear Feature Extraction," Proceedings of ARPA Image Understanding Workshop, Pittsburgh, Pa., Nov. 1978, pp. 73-78.

2. R. Nevatia and K. Price, "Locating Structures in Aerial Images," Proceedings of ARPA Image Understanding Workshop, Palo Alto, Ca., October 1977.

3. K.R. Babu, "Structural Object Recognition in Aerial Images," in Semi-Annual Technical Report, USCIPI 910, R. Nevatia and A.A. Sawchuk, Editors, September 1979.

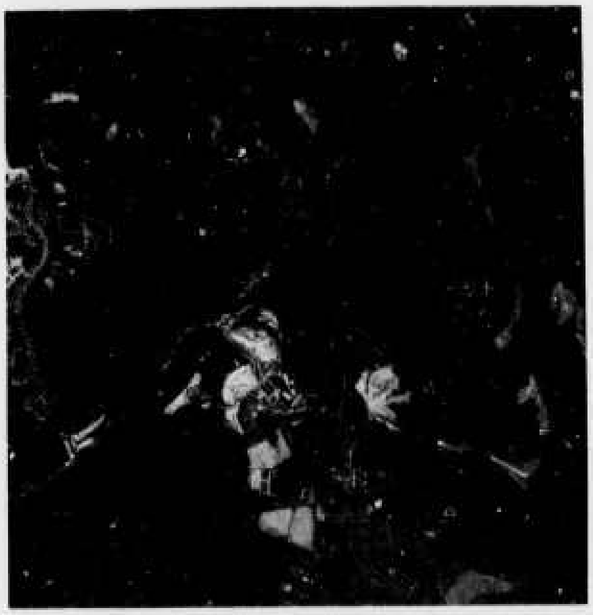


Figure 1. Digitized version of DMA sample image.

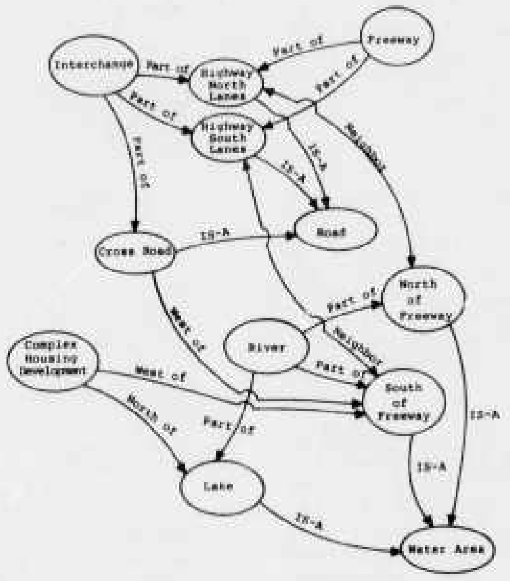


Figure 2. Partial model description for the image in Fig. 1 in terms of a semantic network.

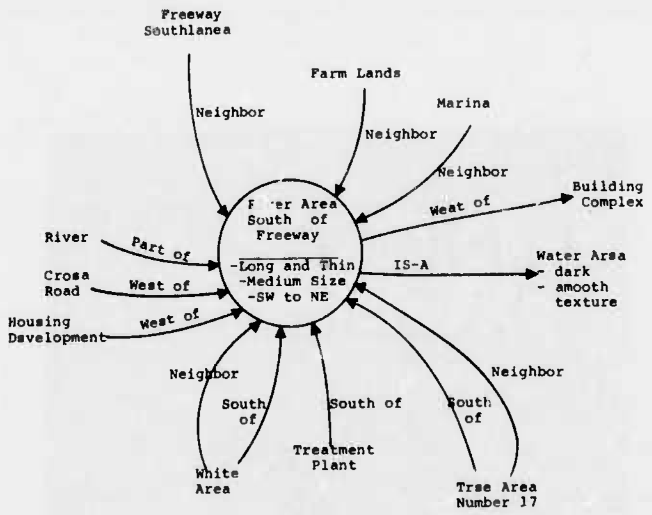


Figure 3. Details of the description of a single node of the network in Fig. 2.

SESSION III

PROGRAM REVIEWS

BY

PRINCIPAL INVESTIGATORS

PROGRESS IN IMAGE UNDERSTANDING RESEARCH AT USC

Ramakant Nevatia
Alexander A. Sawchuk

Image Processing Institute
University of Southern California
Los Angeles, California 90007

Brief summaries of our recent work and of previously developed techniques that may be suitable for use in a demonstration system are provided in this paper.

RECENT RESULTS

In the past six months, research at USC has been directed at the various levels of an Image Understanding system. Only a brief summary is provided here; more details may be found in [1].

Fast Convolution

In the past, Abramatic, Lee and Pratt have investigated convolution with large kernels by using a series of small generating kernels for lower computational cost and possible implementation in inexpensive hardware. Many of these results were presented in a previous report [2]. In recent work, progress has been made in approximating filters by the use of singular value decomposition.

Texture Analysis

We have had several parallel efforts for texture analysis. In theoretical work, Ashjari and Pratt have examined the behavior of singular values of a random texture field. The singular values are believed to be useful texture discriminants. Garber has investigated the use of a mathematical model for analysis and for synthesis of texture. An example of such synthesis is shown in Fig. 1. Figure 1(a) shows a real raffia image which is used to estimate the parameter of the texture field modelled as a Markovian process. Figure 1(b) shows a synthesized raffia texture. The example shown is for a binary image, but the technique has been extended to grey level images.

Laws has devised texture measures computed by small (3x3 or 5x5) filter masks. The classification results appear to be better than for the to more expensive grey level co-occurrence matrix based methods. These measures and results are described in detail in a separate paper in these proceedings [3].

Taking a structural approach to texture, Nevatia, Price and Vilnrotter previously described a technique for generating description of regular textures based on analysis of micro-edges [4]. Further progress has been made in automating the extraction of texture descriptions from analysis of edge repetition counts (or co-occurrences). An example of the types of descriptions obtained for a sample of raffia (similar to, but the same as in Fig. 1) are shown in Fig. 2.

Matching

At the top level of an Image Understanding system, image descriptions need to be matched with stored models for object recognition, change detection, map updating, etc. Previously, we have described a system to match symbolic maps with image descriptions for locating certain desired structures, e.g. airports in aerial images [5]. This system has been modified to incorporate various improvements of the lower level processing. In another project, we are designing a matching system for recognition of structured objects, with initial focus on recognition of airports.

PREVIOUS SIGNIFICANT RESULTS

In this section, we summarize previous results which are believed to be relatively robust and to be considered for inclusion in a demonstration system. It should be noted that even when the techniques have been tested on a large number of images, their performance for particular applications, e.g. DMA applications, remains to be evaluated.

Linear Feature Extraction

We have developed a technique for linear feature extraction that operates by deleting edges using small masks, thinning the resulting edges, and linking them to give boundary segments [6]. Linear feature are important for man-made and natural objects. Our technique has been applied to the detection of objects such as roads and airport runways, but is not specially designed for detection of such features. This technique is also being used at Hughes Research Laboratories and by T. Binford at Stanford University for other applications.

Image to Map Correspondence

This technique has been described previously [5], and is potentially useful for map guided change detection, map updating and guidance. We have applied it to aerial images of diverse areas in California such as San Francisco, Stockton, and San Diego. However, the basic matching technique is simple and its performance on a larger set of images with well defined goals needs to be evaluated. Our technique uses the above described linear feature extraction technique and the Ohlander-Reddy-Price segmentor developed at CMU.

Texture Measures

We have developed a number of texture measures in the past [3,7]. These measures have been tested primarily on natural textures in Brodatz's album [8]. They should be effective in classification of materials in aerial images and also possibly for image segmentation. However, their performance on range of textures and materials encountered in aerial images used for map-making remains to be tested.

REFERENCES

1. R. Nevatia and A.A. Sawchuk, (Editors) Semi-Annual Technical Report, USCIPI #910, October 1979.

2. W.K. Pratt, J.F. Abramatic and S.U. Lee, "Two-dimensional Small Generating Kernel Convolution," in USCIPI Report #860, March 1979.

3. K. Laws, "Texture Energy Measures," Proceedings ARPA Image Understanding Workshop, Los Angeles, Ca., 1979 (these proceedings).

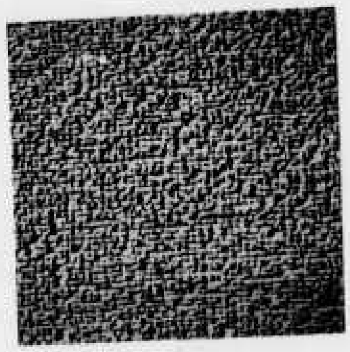
4. R. Nevatia, K. Price and F. Vilnrotter, "Describing Natural Textures," Proceedings of ARPA Image Understanding Workshop, Palo Alto, Ca. April 1979, pp. 55-60.

5. R. Nevatia and K. Price, "Locating Structures in Aerial Images," Proceedings of ARPA Image Understanding Workshop, Palo Alto, Ca., October 1977.

6. R. Nevatia and K.R. Babu, "Linear Feature Extraction," Proceedings of ARPA Image Understanding Workshop, Pittsburgh, Pa., Nov. 1978, pp. 73-78.

7. W.K. Pratt and O.D. Faugeras, "Decorrelation Methods of Texture Feature Extraction," in Semiannual Technical Report, USCIPI #860, H.C. Andrews and W.K. Pratt, Editors, March 1979.

8. P. Brodatz, Textures, A Photograph Album for Artists and Designers, New York: Dover 1966.


a) Raffia Image

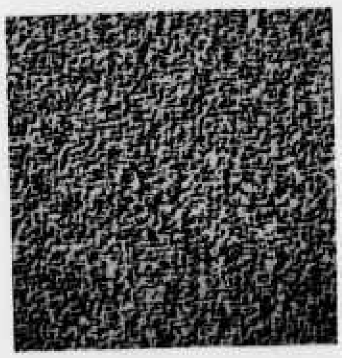

b) Synthesized Raffia Texture.

Figure 1. A raffia image and synthesized texture.

```
DARK OBJECT DESCRIPTIONS

HORIZONTAL SCAN DIRECTION

     THERE IS STRONG EVIDENCE OF PERIODICITY WITH ELEMENT SPACING 11.00000

     THERE IS STRONG EVIDENCE OF PERIODICITY WITH ELEMENT SPACING 12.00000

     THERE IS WEAK EVIDENCE OF ELEMENT SIZE 10.00000
     WITH STRONG SUPPORT FOR ELEMENT SPACING 11.00000

45 DEGREE SCAN DIRECTION

     NO EVIDENCE OF PERIODICITY OR PREDOMINANT ELEMENT SIZE

VERTICAL SCAN DIRECTION

     THERE IS VERY STRONG EVIDENCE OF PERIODICITY WITH ELEMENT SPACING 8.000000

     THERE IS STRONG EVIDENCE OF ELEMENT SIZE 2.000000
     WITH STRONG SUPPORT FOR ELEMENT SPACING 8.000000

135 DEGREE SCAN DIRECTION

     NO EVIDENCE OF PERIODICITY OR PREDOMINANT ELEMENT SIZE


LIGHT OBJECT DESCRIPTIONS

HORIZONTAL SCAN DIRECTION

     NO EVIDENCE OF PERIODICITY
     MODERATE EVIDENCE OF ELEMENT SIZE OF 2.000000

45 DEGREE SCAN DIRECTION

     NO EVIDENCE OF PERIODICITY OR PREDOMINANT ELEMENT SIZE

VERTICAL SCAN DIRECTION

     THERE IS STRONG EVIDENCE OF PERIODICITY WITH ELEMENT SPACING 8.000000

     THERE IS STRONG EVIDENCE OF ELEMENT SIZE 6.000000
     WITH STRONG SUPPORT FOR ELEMENT SPACING 8.000000

135 DEGREE SCAN DIRECTION

     NO EVIDENCE OF PERIODICITY OR PREDOMINANT ELEMENT SIZE
```

Figure 2. Structural machine description of a raffia texture.

THE SRI IMAGE UNDERSTANDING PROGRAM

By M. A. Fischler
(Principal Investigator)
SRI International
Menlo Park, California

INTRODUCTION

Research at SRI International under the ARPA Image Understanding Program was initiated to investigate ways in which diverse sources of knowledge might be brought to bear on the problem of analyzing and interpreting aerial images. The initial phase of research was exploratory and identified various means for exploiting knowledge in processing aerial photographs for such military applications as cartography, intelligence, weapon guidance, and targeting. A key concept is the use of a generalized digital map to guide the process of image analysis. The results of this earlier work were integrated in an interactive computer system called "Hawkeye" [1]. This system provides basic facilities necessary for a wide range of tasks in cartography and photo interpretation, and it provides a framework within which other applications can be readily demonstrated.

Research has recently been focused on development of a program capable of expert performance in a specific task domain--road monitoring. This work, now reaching a test and evaluation phase, is described in the following section, "The SRI Road Expert."

We are currently initiating major efforts in two new directions. The first is in support of a joint ARPA/DMA program to provide a framework for demonstrating the applicability of image understanding research (from throughout the entire IU community) to military problems in general, and to the problems of automated cartography in particular. Our plans for this effort are described later in this paper.

The second effort is to broaden the scope and generality of our current work concerned with analyzing aerial imagery--specifically in the areas of 3-D terrain understanding, perceptual reasoning, and image description and matching. A separate research program (described in the last section of this paper), jointly supported by ARPA and NSF, will augment these investigations by attempting to clarify the computational principles underlying the early stages of visual processing in men and machines.

THE SRI ROAD EXPERT

The primary objective of our research concerned with development of the SRI Road Expert is to build a computer system that "understands" the nature of roads and road events. It is intended to be capable of performing such tasks as:

(1) Finding roads in aerial imagery.

(2) Distinguishing vehicles on roads from shadows, signposts, road markings, etc.

(3) Comparing multiple images and symbolic information pertaining to the same road segment, and deciding whether significant changes have occured.

The system should be capable of performing the above tasks, even when the roads are partially occluded by clouds or terrain features, are viewed from arbitrary angles and distances, or pass through a variety of terrains.

The general approach and details of technical progress on developing the components of the Road Expert are contained in References [2-6]. We are currently integrating these separate components into a coherent system that facilitates testing and evaluation and will be in a form suitable for transfer to the ARPA/DMA Integrated Demonstration System ("testbed"). Plans for the Road Expert Demonstration System, expected to be completed early in 1980, are presented below.

The Road Expert Demonstration System

Given an image and a description of the approximate circumstances under which it was taken, the road expert determines a precise geometric correspondence between the picture and a model of the world, uses this correspondence to predict the positions of the roads to be monitored, scans along the roads for possible vehicles, and classifies the vehicles on the basis of their size, shape, and shadow structure.

Initialization Procedure

The user can select one of more than 20 currently available pictures from our library or provide one of his own; the picture will then be scanned ("on-line") by a TV camera, which digitizes a 512 x 512 portion of it. To test performance under adversity, the user can superimpose clouds and their shadows on the scanned image. Clouds are added by digitizing a portion of a cloud picture and can be interactively positioned in any desired location in the test image.

The required description of the test picture includes the following information:

* The date and time the picture was taken
* The internal geometry of the camera that took it
* An estimate of the camera's position and orientation when the picture was taken
* An estimate of the uncertainties associated with the estimates of the camera's position and orientation

In a real application the internal camera model would be known a priori, and the rest of the information would be obtained directly from the platform's navigation system (e.g., inertial navigation). In the context of the demonstration system, the user has to supply this information or compute it using interactive aids included in the system. The date and time the picture was acquired is known (or has been estimated from shadow information) for all the images in our data base. The internal camera parameters (such as focal length) are either known or are computed as part of the image-to-data-base correspondence process (see below). External camera parameters (position and orientation, in world coordinates) have been precomputed for all images in the library and stored in the system's data base. We allow the user to perturb these estimates and to specify uncertainties in order to simulate the effects of navigation errors on system performance.

Scale factors and displacements associated with image digitization, which would be known in an operational setting, are obtained by an interactive procedure requiring the user to select corresponding points on a contact print of the original image (which is mounted of an xy digitizing tablet) and on the digitized image (which is displayed on a television screen).

Analysis

The first task in the analysis is to improve the estimates of the camera's real-world location and orientation. The system uses the initial parameter values to predict landmarks that are likely to appear in the picture. According to a preplanned strategy, the system tries to locate the linear landmarks first and compute an improved set of parameters. The new parameter values are used to find point landmarks, and a final set of parameter values are computed on the basis of all the located landmarks.

Given the new improved set of camera parameter values, the system constructs a display that shows the predicted positions of the roads to be monitored. These predictions are used to guide the road tracker, which scans the roads for possible vehicles. The road tracker marks each candidate (road anomaly) on the display and passes it to the vehicle recognition subsystem for analysis.

The recognition subsystem first tries to determine if the candidate is a road marking, a cloud, a shadow, or a vehicle. A candidate is declared to be a road marking if it occurs at the position of a known marking. A candidate is declared to be a cloud if it is one of the brightest regions in the picture, relatively large, and has a low internal brightness variance. If a candidate is one of the darkest regions in the picture and it extends to the side of the road on which the sun is located, it is assumed to be the shadow of a tall object off the road. If the candidate is not one of the above, has the right size and shape, and has a shadow that is consistent with the sun's location, it is called a vehicle. If a candidate does not satisfy any of these descriptions, it is labeled as an unrecognized anomaly.

All steps in the above demonstration sequence are accompanied by appropriate displays, and facilities exist for user interrogation of system-produced intermediate results.

THE ARPA/DMA INTEGRATED DEMONSTRATION SYSTEM ("TESTBED")

ARPA and DMA have jointly agreed to establish an integrated demonstration system ("testbed"), with SRI as the integrating contractor. The system, which is to be located at SRI, is intended to be used for demonstrating and evaluating the applicability of IU research to cartography. For this purpose it will have a user interface that simulates the environment of a cartographic work station, consisting of a computer with CRT terminal, an image display with track ball, and a digitizing tablet. With the exception of image digitization, which for most purposes will be performed off-line by DMA, the system will support all major steps in map making with a continuously evolving degree of automation.

Initially the system will allow interactive creation and editing of digital maps in a fashion similar to Hawkeye [1]. Existing maps, for example, can be overlaid on new imagery, edited, and extended, using a variety of interactive aids for tracing linear features and modeling objects. The system will also allow remote execution (over the ARPANET) of automated and semi-automated techniques developed by IU contractors. Those techniques whose utility and reliability justify tighter integration will then be incorporated into the system.

The system will also be usable as a research facility, providing the IU and DMA communities with access to the most advanced tools available. This, plus the existence of compatibility standards for programs and data, will encourage building upon the work of others, allowing more ambitious projects to be undertaken. Furthermore, the availability of common data sets will [illegible]te more systematic evaluation of competing techniques.

The above objectives share the requirement for a very flexible architecture so that contributions developed in a diverse community can be fully utilized. Integration support must be provided for a wide range of contributions, from primitive image processing techniques (e.g., an edge follower) to stand-alone subsystems (e.g., an expert system for terrain modeling). Language and operating system support must be sufficiently broad to encompass programs running under the multitude of systems used throughout the community. These systems can be expected to change continuously during the testbed's lifetime.

To meet these requirements, the system will be configured as a library of application modules that accepts input and deposits results in a shared global data base. For normal research and development, modules can be directly controlled in an interactive environment via the keyboard and graphical devices. For demonstrations a front-end process is interposed, simulating the environment of a cartographic work station similar to Hawkeye. This interface facilitates communication with the system via menus (e.g., ZOG) or limited natural language (e.g., LIFER, RITA) and includes a "help" facility.

The application modules each perform a well-defined high- or low-level task. Modules are independently compiled so each can be implemented in different languages and can reside on different processors. Modules interact with each other by means of a standard interfacing mechanism, resembling a procedure call. Details of how control is actually passed will, of course, vary, depending upon the level of module integration (same address space, same processor, or remote processor).

Most data interactions between modules will be affected by accessing the shared data base. Modules operate on common data from the data base and deposit their results back into the data base, where they will be available for display or subsequent processing by other modules. The data base thus acts like a blackboard for interprocess communication in a fashion similar to Hearsay II [7].

The data base is accessed via a uniform query language. This enforces compatibility and maintains integrity without constraining a module's internal representation. Modules need not know the source of the data they use nor who will use their results. For example, a program that needs the locations of edges in image X will look in the data base; if they are not there, then the program requests the use of an edge-locator module which will deposit results, tagged as "edges for image X" in the data base, where they will remain available for future use. The data base is thus the key to modularity in a large integrated system.

The VAX 11/780 has been selected as the main testbed machine. All tightly integrated parts of the system will be resident there. All other IU machines will be viewed uniformly as remote ARPANET hosts, including SRI's KL-10. However, the SRI KL will have a high-speed channel to the VAX via a shared disk, so application modules running there will incur minimal overhead.

SRI will build a core system on the VAX. It will include a data base and some general system utilities, such as display servers, data base manager, work station front end, and an ARPANET gateway.

Development of application modules will proceed in parallel at all IU sites. Each site will maintain local copies of relevant parts of the official data base (e.g., imagery) needed to develop and demonstrate their routines. Each site will also be able to call remotely resident modules over the network, using the gateway mechanism.

SRI will use the ARPANET to exercise application modules in the context of the core system. As utility and performance justify, modules will be imported to run at SRI on our KL or VAX. As the final demonstration takes form, critical modules may be recoded to integrate efficiently with the core VAX environment.

Our time schedule is to complete definition of the system by the end of this year, do the detailed design and construction of the core system in 1980, integrate application modules provided by the IU community in 1981, and evaluate and possibly extend the system in 1982.

RESEARCH INTO COMPUTATIONAL PRINCIPLES OF VISION

A major thrust of basic vision research at SRI, now jointly supported by ARPA and NSF, is to understand the computational principles underlying early stages of visual processing in men and machines. Our previous research strongly suggested that an important function of early vision is the transformation of gray-level information in the input image into an intermediate level of representation that describes the intrinsic characteristics (e.g., depth, orientation, reflectance, color, incident illumination, and so on) of the three-dimensional surface element visible at each point in the image [8]. Such characteristics are important in their own right (e.g., for describing terrain surfaces) and are also fundamental to higher levels of perception (such as object recognition).

The central problem in recovering intrinsic surface characteristics is that the needed information is confounded in the single value of light intensity available at each point in the image. Recovery thus depends upon constraints derived from assumptions about the nature of the scene and the physics of the imaging process (e.g., surfaces are smooth and are viewed from a general position). There are many techniques for recovering particular types of information, given certain specific circumstances; an important part of the recovery process is to decide which techniques are applicable in which areas of the image and how to integrate the results to obtain a consistent overall interpretation. A theoretical model for simultaneously recovering orientation, reflectance, and illumination images from a single monochrome image was developed and is currently being implemented.

A major component of the implementation is a technique for interpolating surface shape from sparse, possibly conflicting, evidence and constraints concerning orientation and range. Such interpolation is a pervasive problem in low-level vision, arising, for example, in generating surface shells from stereo disparities or from shading information and boundary conditions.

In the context of the above model, we plan to use interpolation to infer surface shape in regions where analytic photometry cannot be used to derive shape from shading (as is the case when illumination is too uniform or too complex for adequate modeling). In such cases the primary source of information is the discontinuities in the image which correspond to surface boundaries (as are usually represented in line drawings). We are attempting to understand how line drawings convey three-dimensional surface structure. The problem is again one of ambiguity, since each line in the image represents an infinitude of possible three-dimensional space curves; the solution again requires exploiting assumptions (smoothness of surfaces and generality of viewpoint). We have implemented several computational models for recovering the three-dimensional shape of a space curve from an image curve. This provides partial constraints on the range and orientation of the surface bounded by that curve. We are currently integrating the curve interpretation and interpolation processes into a system that can interpret simple line drawings as three-dimensional surfaces.

ACKNOWLEDGEMENT

Major contributors to the SRI Image Understanding Program are: C. J. Agin, H. G. Barrow, R. C. Bolles, M. A. Fischler, T. D. Garvey, G. A. Jirak, D. L. Kashton, J. M. Tenenbaum, and H. C. Wolf.

REFERENCES

1. H. G. Barrow et al., "Interactive Aids for Cartography and Photo Interpretation: Progress Report, October 1977," in Proceedings: Image Understanding Workshop, pp. 111-127 (October 1977).

2. M. A. Fischler et al., "Interactive Aids for Cartography and Photo Interpretation," Semiannual Technical Report, SRI Project 5300, SRI International, Menlo Park, California (October 1978 and May 1979).

3. L. Quam, "Road Tracking and Anomaly Detection," in Proceedings: Image Understanding Workshop, pp. 51-55 (May 1978).

4. R. C. Bolles et al., "The SRI Road Expert: Image-to-Database Correspondence," in Proceedings: Image Understanding Workshop, pp. 163-174 (November 1978).

5. C. J. Agin, "Knowledge-Based Detection and Classification of Vehicles and Other Objects in Aerial Road Images," in Proceedings: Image Understanding Workshop, pp. 66-71 (April 1979).

6. M. A. Fischler, J. M. Tenenbaum, and H. C. Wolf, "Detection of Roads and Linear Structures in Aerial Imagery," in Proceedings: Image Understanding Workshop (November 1979).

7. D. R. Reddy et al., "Speech Understanding Systems," Final Report, Computer Science Department, Carnegie-Mellon University, Pittsburgh, Pennsylvania (1977).

8. H. G. Barrow and J. M. Tenenbaum, "Recovering Intrinsic Scene Characteristics from Images," in Computer Vision Systems, A. R. Hanson and E. M. Riseman, eds. (Academic Press, 1978).

Recent Developments in the Rochester Image Understanding Project

J. A. Feldman
K. R. Sloan, Jr.

The University of Rochester
Rochester, New York 14627

1. Model Refinement

1.1. Constraint Networks and Procedural Description

One important goal of the Rochester Vision Project is to investigate a generalized representation of complex objects by semantic networks. In our formulation these include procedural invocation in which an executive procedure chooses worker procedures to perform a job not just on the basis of input/output behavior (as traditional pattern-directed invocation does), but also taking into account cost/benefit estimates and perhaps other information as well. This scheme is motivated by the desire to have the advantages of declarative knowledge about what is doable (the descriptions) along with the advantages of procedural knowledge about how to do it (the workers). The declarative, descriptive component will allow conviences such as the modular addition of procedural knowledge. The main research issue is to decide what exactly needs to be known about worker procedures, and how to express that in a useful and uniform manner. This must also be coordinated with the use of relational constraints [Russell and Brown, 1978]. A recent paper at Rochester exploring aspects of these issues is [Lantz, Ballard, and Brown, 1978].

1.2. Decision Theory

The use of decision theory not only as an abstract model of intelligent perception but as a practical tool to maximize computational benefit/cost is being investigated in the context of procedural invocation. This work continues in the tradition of Bolles, Sproull, and Garvey, and ultimately we hope to extend some of their results to deal with formal problems that more closely approximate the sorts of vision problems encountered in our particular applications. Ballard (see Section 8) uses decision theory techniques to choose the most economical method (assuring adequate accuracy) of locating anatomical structures in large-format images.

2. Noise-Resistant Feature Detection

Work has been underway at Rochester for several years on developing techniques for reliably detecting specific visual features, even in the presence of considerable noise. Our work has been based on generalizations of the Hough technique, which accumulates evidence for straight lines at various slope and intercept values using an accumulator array. For some time, we have been successfully employing extended Hough techniques to locate second-order curves like elliptical sections and circles. In the last six months we have been able to extend these techniques to handle a broad class of features [Ballard, 1979a]. There is reason to believe that these noise-resistant feature identification methods can be combined with our constraint graph techniques (cf. Section 1) to yield a robust and general analyzer for industrial site images.

3. Application in Aerial Image Analysis

The three-level organization of image analysis (strategist, executive, worker) and a further exploration of useful procedural description mechanisms were first applied to photointerpretation work in [Lantz et al., 1978]. The object is to use the sorts of knowledge-based inferencing used by skilled photointerpreters, along with models inspired by photointerpretation keys for identifying small industries, to do reliable and flexible identification of a few types of small industrial installations.

A second phase of experimentation was based on the analysis of selected industrial sites using locally acquired aerial imagery. We have now acquired and digitized a sample image from the Defense Mapping Agency and are working on the structure of our third generation system. The current plan is to rely heavily on the general techniques described in Sections 1 and 2 above.

4. Image Encoding and Transmission

4.1 Hierarchical Image Encodings

Communication of images, and information about images is an important part of any image understanding project. We have been investigating the use of various hierarchical image encodings. One of the image transmission schemes we have investigated is closely related to "pyramid" data structures. We have demonstrated that high resolution raster images can be effectively transmitted over relatively low-bandwidth lines by sending a series of low resolution approximations, which converge to the final image [Sloan and Tanimoto, 1978].

A second hierarchical encoding termed Strip Trees has been described elsewhere [Ballard, 1979b]. A Strip Tree is an elegant encoding for curves which represents both open curves (linear features) and closed curves (areas) in a uniform manner. Strip Trees are closed under intersection and union and operations on them can be carried out at different resolutions. These properties make them a nearly ideal representation for map data bases.

4.2 Composition and Re-interpretation of Images

It is often convenient to specify an image in terms of the combination of several existing images, rather than transmit an entire new image. The combination or re-interpretation may sometimes be performed with relatively simple hardware devices. We have developed and implemented several such techniques based on the "video lookup table" supplied with our Grinnell GMR-26 display [Sloan and Brown, 1979]. These techniques are currently being used to overlay map features on aerial images, display three-dimensional surfaces under quickly varying lighting conditions, and show short, repetitive motion sequences.

5. Component Building

5.1. Hardware

The Grinnell GMR-26 display device is DMA-interfaced to an Eclipse computer, and has been invaluable as an output device for our experiments. An Optronics Colorscan C-4100 drum scanner is on site and interfaced to the Vision Eclipse. The fast (50KB) link to the PDP-KL10 has been completed and is operating well.

Both Eclipse computers are fully configured and have been running effectively with our distributed system software. A VAX 11/780 (purchased with non-DOD funds) is operating and will be integrated into the local network. We are acquiring terminals and investigating how to meet our everyday computing needs by commercial, home-built, or combination intelligent terminal systems.

5.2. Software

Advanced system software support is now used routinely, and more is under development. Communications protocols and distributed computing packages [Feldman, 1978; Sheininger and Sabbah, 1977; Selfridge, 1979; Sloan, 1978] have been developed to allow access to the GMR-26 through the local ALTO computers or the remote PDP-10, to achieve reliable transmission between distributed processes, to produce graphics and halftone images on ALTO screens from the PDP-10, and to allow file transfer and telnet to the Arpanet. At Rochester, the RIG message is the lingua franca that allows processes on remote machines to command the GMR-26, perform file manipulations, and other operations. Some of our work has been utilized by other image understanding groups, most extensively at SRI. We are actively engaged in cooperative efforts with other DARPA contractors to develop a general and flexible set of software tools.

A comprehensive library of vision routines [Sloan, 1977-78] has been developed, centralized, documented, and incorporated into the NEXUS system. They allow interactive users a wide range of image-processing and display (graphics, halftone, color and B&W TV) capabilities. A program to acquire images from the Optronics scanner and package them according to our Raster Image File Format [Selfridge and Sloan, 1979] has been developed and is in routine use.

6. Motion Understanding

Understanding motion pictures has always presented an unusually difficult problem to computer vision efforts. The compelling gestalt induced in humans by moving objects is not well understood, and so there is little leverage on the immediate problems resulting from the large mass of data in multi-frame images. We began on a pared-down version of the problem which nevertheless offers an interesting set of perceptual phenomena to model. The domain is multi-frame images of animal motion; initial research is being carried out on sequential images of points of light attached to joints. (A detailed progress report is presented in these Proceedings.)

7. Texture

Textural areas can be thought of as those parts of an image where segmentation based on normal similarity measures fails. Meaningful analysis of textured areas must include discrimination between different textures and detection of parts of the same texture. The similarity of textures which are identical except for a scale change, a rotation, or a different range of intensities must be recognized.

We approach the texture problem by dividing texture regions into meaningful sub-elements of similar intensity sample points, then using rotation- and scale-invariant shape measures to characterize these regions and finally determining spatial relationships among our sub-elements. By using a decision tree program structure, easily discriminated textures are separated quickly, and more complex textural structure is extracted only when necessary [Maleson, Brown, and Feldman, 1977].

8. Applications in Biomedicine

The model-directed finding of ribs in chest radiographs [Ballard, 1978] provides an illustration of the use of the Rochester Vision System, incorporating procedure description, utility measures, and tops-down, model-directed perception. The object here is to cope with large amounts of possibly low-quality data without undue processing time by depending on a declarative model of anatomical structures, described procedural knowledge about how to locate them, and an executive which uses decision theory to control the image-understanding process. A prototype complete analysis system is now being developed.

A novel and uniform method of describing arbitrary functions on the unit sphere (which define "museum-viewable" volumes) is under investigation, with immediate application to anatomical structures [Schudy and Ballard, 1979]. The idea is related to the well-known Fourier descriptions of two-dimensional shape. Volumes are modelled and described as the leading coefficients in certain spherical harmonic expansions of the volume functions. This method also allows least squared error fitting of volumes in coefficient space, which interfaces nicely with routines which locate the three-dimensional boundaries of volumes in image data.

Applications of generalized cylinders [Agin, 1972] previously have been limited to simple cross sections. We use B-splines as an embedding for generalized cylinders [Shani, 1979]. This allows an efficient realization of the original notion of generalized cylinders as arbitrary cross sections about a space curve.

REFERENCES

Agin, A.P., Representation and description of curved objects, Stanford AI Project Memo AIM-173 (and Ph.D. thesis), October 1972.

Ballard, D.H., Model-directed detection of ribs in chest radiographs, TR11, Computer Science Department, University of Rochester, March 1978.

Ballard, D.H., Generalizing the Hough Transform to detect arbitrary shapes, TR55, Computer Science Department, University of Rochester, October 1979 (a); submitted to Pattern Recognition.

Ballard, D.H., Strip Trees: A hierarchical representation for map features, Proc. IEEE Conf. on Pattern Recognition and Image Processing, Chicago, Illinois, August 1979 (b).

Barrow, H.G., et al., Interactive aids for cartography and photo interpretation, Semiannual Technical Report, Artificial Intelligence Center, SRI International, November 1977.

Feldman, J.A., Synchronizing distant cooperating processes, TR26, Computer Science Department, University of Rochester, October, 1977.

Feldman, J.A., Systems support for advanced image understanding. DARPA Semiannual Technical Report, May 1978.

Feldman, J.A., and Williams, G. Some comments on datatypes, TR28, Computer Science Department, University of Rochester, December 1977.

Lantz, K.A., Brown, C.M. and Ballard, D.H., Model-driven vision using procedure description: Motivation and application to photointerpretation and medical diagnosis, 22nd International Symposium of the Society of Photo-optical Instrumentation Engineers, San Diego, CA., August 1978.

Maleson, J.T., Brown, C.M. and Feldman, J.A., Understanding natural texture, DARPA Image Understanding Workshop, October 1977.

Russell, D.M., Where do I look now?: Modeling and inferring object locations by constraints, Proc. IEEE Conf. on Pattern Recognition and Image Processing, Chicago, Illinois, August 1979.

Russell, D.R., and C.M. Brown, Representing and using locational constraints in aerial imagery, Image Understanding Workshop, November, 1978.

Scheininger, U., and Sabbah, D., The display process, Internal Memo, Computer Science Department, University of Rochester, December 1977.

Schudy, R.G. and Ballard, D.H., Towards an anatomical model of heart motion as seen in 4-d cardiac ultrasound data, IEEE Conf. on Computer-Aided Analysis of Radiological Images, June 1979.

Selfridge, P.G., A flexible data structure for accessory image information, TR45, Computer Science Department, Universiiy of Rochester, May 1979.

Selfridge, P.G. and Sloan, Jr., K.R., Raster image file format (RIFF): An approach to problems in image management, Proc. IEEE Conf. on Pattern Recognition and Image Processing, Chicago, Illinois, August 1979.

Sloan, Jr., K.R., Rochester vision library documentation, Internal Memos, Computer Science Department, University of Rochester, 1977-1978.

Sloan, Jr., K.R., and Brown, C.M., Color map techniques, Computer Graphics and Image Processing 10, 4, August 1979.

Sloan, Jr., K.R., and Tanimoto, S.L., Progressive refinement of raster images, TR39, Computer Science Department, University of Rochester, November 1978.

Shani, U., B-splines as an embedding for generalized cylinders, Computer Science Department, University of Rochester, forthcoming.

MIT IMAGE UNDERSTANDING TECHNIQUES

Patrick H. Winston

The Artificial Intelligence Laboratory
Massachusetts Institute of Technology

In the November, 1978 Proceedings, we reviewed the overall program, briefly explaining our approach, stating the objectives, and citing the fundamental tools. Then we summarized the results obtained through an enumeration of representative individual efforts.

In the April, 1979 Proceedings, we concentrated on Horn's group's work on hill shading *and* atmospheric modeling *and on Marr's group's discoveries about* texture *and about* zero crossings, *with particular emphasis on* stereo.

Here we confine ourselves to a brief enumeration of those techniques that are most ready for testbed experiments, ignoring all work that is further up the research and development pipeline. All things listed have been described at length in previous workshop proceedings.

Shaded Images using Reflectance Maps

Shaded images give a good, immediate impression of surface topography. They are, however, expensive to make when done by a human artist with an air brush.

The solution is to make synthetic images using a *reflectance map* to capture and tabulate the constraint that determines image intensity from sun position, viewer position, surface orientation, and surface material. The steps are as follows:

* Use a terrain model to compute local normals.

* Use the local normals to find the proper intensity in the reflectance map. Use a reflectance map that best matches the surface material if the image is to be realistic or use one that gives the best depth impression if the image is to be used as a map overlay.

This subject was dealt with at length in Horn's article in the April, 1979 Proceedings. Among other things, he reviews the important hill-shading work going back to the maps of Lenardo da Vinci drawn in 1502 and 1503. He also points out that the necessary reflectance map can be computed from the bidirectional reflectance distribution function (BRDF) using techniques described by Horn and Sjoberg in the November, 1978 Proceedings.

Image Registration using Synthetic Images

Ground control points or other sharply defined features may not be available for registering images with terrain models or with one another.

One alternative to the use of ground control points is to match the given image against a *synthetic image* made from the terrain model. The steps are as follows:

* Make a synthetic image using the terrain model.

* Blur both the synthetic image and the real image.

* Correlate the two images. Find values for transformation parameters that give correlation maxima. The blurring is necessary to eliminate confusing secondary peaks in the correlations.

* Repeat with less and less blurring using accumulated results to limit the search domain.

Subpixel accuracy has been obtained in registering a sample image. Figure 1 illustrates the reduction of images that is one of the key steps in the process.

This basic technique was described by Horn and Bachman in the October, 1977 Proceedings. Since that time, improvements have made it possible to deal with six degrees of freedom, not just the four worked with originally.

Destriping Satellite Images

Before registration and other image-oriented operations can proceed, it is necessary to do quite a lot to clean up and transform the raw satellite image. Getting rid of the stripes that

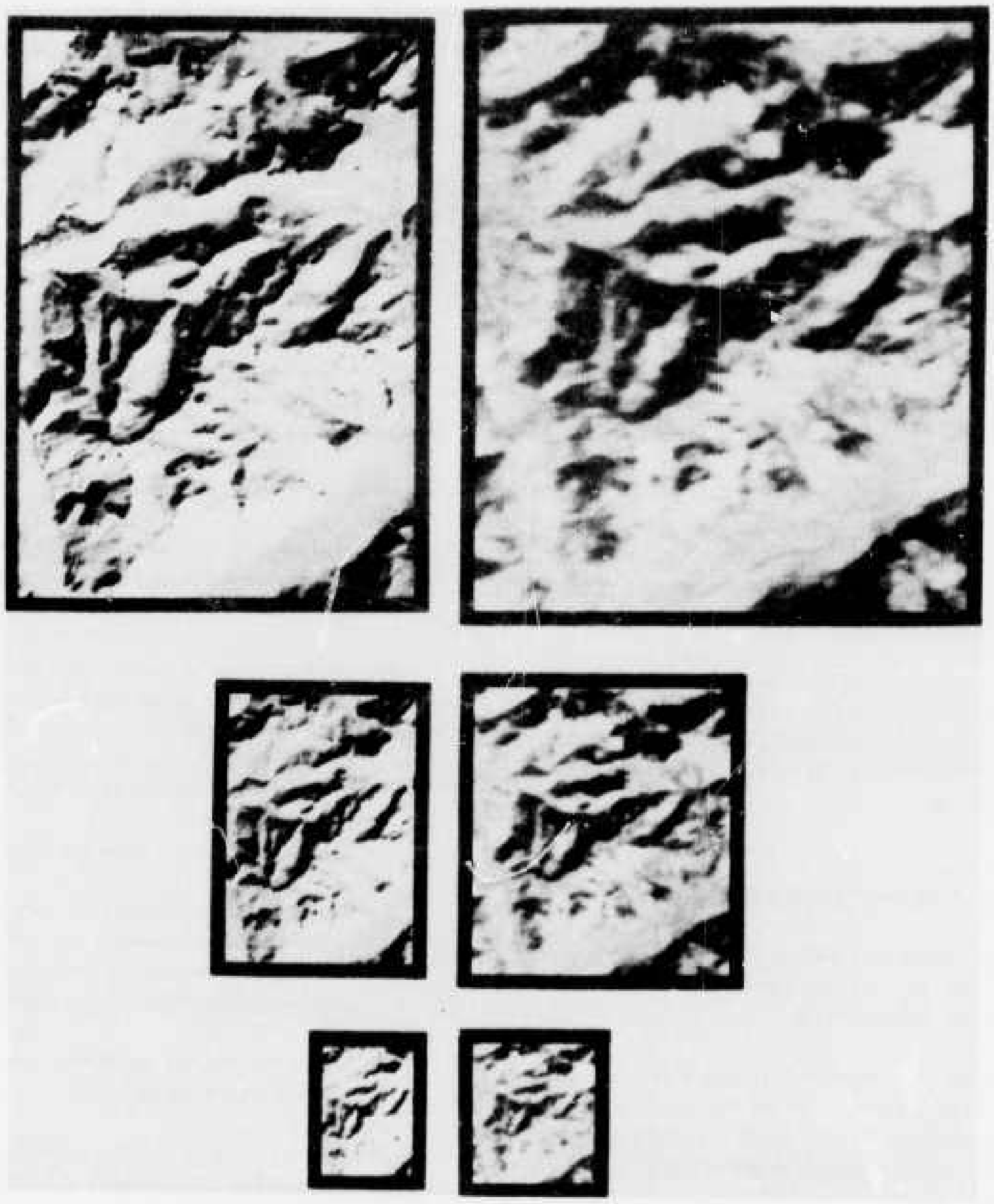


Figure 1 Registering images with digital terrain models requires matching blurred images first. These blurred images are called reduced images.

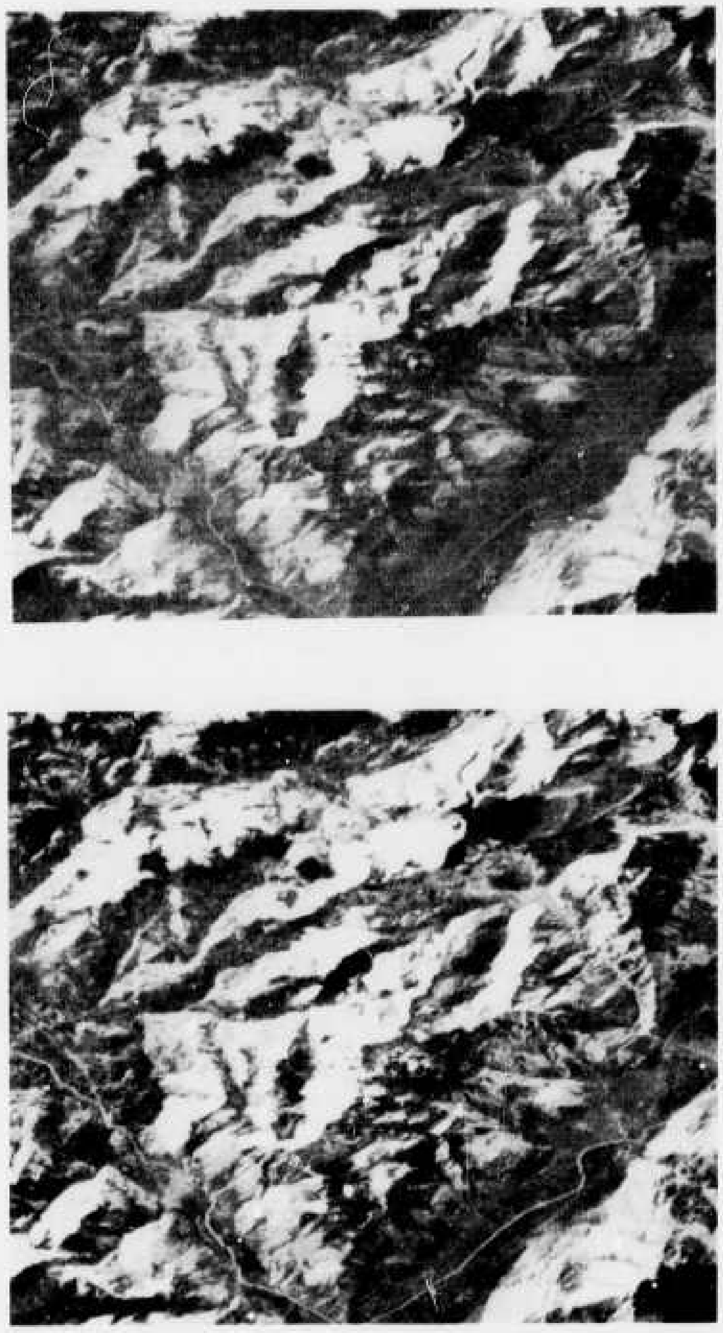


Figure 2: Satellite images typically have unacceptable striping effects. Destriping can be done using histogram comparisons.

are the result of using six independent sensors is a prominent example. This can be done by assuming that all six sensors see the same distribution of intensity values and then using the following steps:

* Make histograms of the intensities seen by each individual sensor and a general histogram of the intensities of all sensors together.

* Create a correction table for the output of each individual sensor by appropriately comparing its histogram with that of the general histogram.

Figure 2 shows an image before and after destriping. The reproduction of these proceedings may not be adequate to show the severe striping in the unprocessed image.

This technique was describe in detail by Horn and Woodham in the May, 1978 Proceedings.

Ground-cover Analysis using Albedo Images

The amount of light reflected by a surface is influenced by the material it is made of and by its orientation. Consequently, multispectral pattern-recognition methods do not work well in hilly terrain where the varying orientation is a factor.

The solution to the problem is to compute an *albedo image* in which only surface material influences intensity. An albedo image is computed as follows:

* Compute a synthetic image using a terrain model. Assume the same sun position that applies to the real image to be analyzed.

* Register the synthetic image with the real image. Use Horn's registration method.

* Divide the intensity at each point in the real image by the corresponding, predicted intensity in the synthetic image.

The result is a flattened image. Figure 3 illustrates. Such images are ready for study in an interactive environment with a human interpreter. Further work is underway to improve albedo images by using atmospheric models to create better synthetic images.

Our plan is to describe this work in the next volume of the proceedings.

Depth Mapping using Stereo

Stereo image analysis is surprisingly hard. Straightforward correlation of images is difficult for two reasons; first, appearance is strongly influenced by viewing angle; and second, correlation requires enormous computation.

One solution is to use *zero-crossing stereo*. This involves matching zero-crossings in image derivatives. The steps are as follows:

* Blur the image by convolution with a Gaussian filter. The most blurred image starts the process by giving rough results that are sharpened by two or three less-blurred images later.

* Apply a Laplacian operation.

* Find zero-crossings in the result. Theory hints, curiously, that the image can be recovered from the zero crossings. Here, they serve to find and pinpoint edges.

* Match the zero crossings in one image with those in the other. Match closest zero crossings that cross in the same direction.

* Repeat using less blurring. Use results accumulated so far to limit the search for matching zero crossings.

Figure 4 illustrates the process.

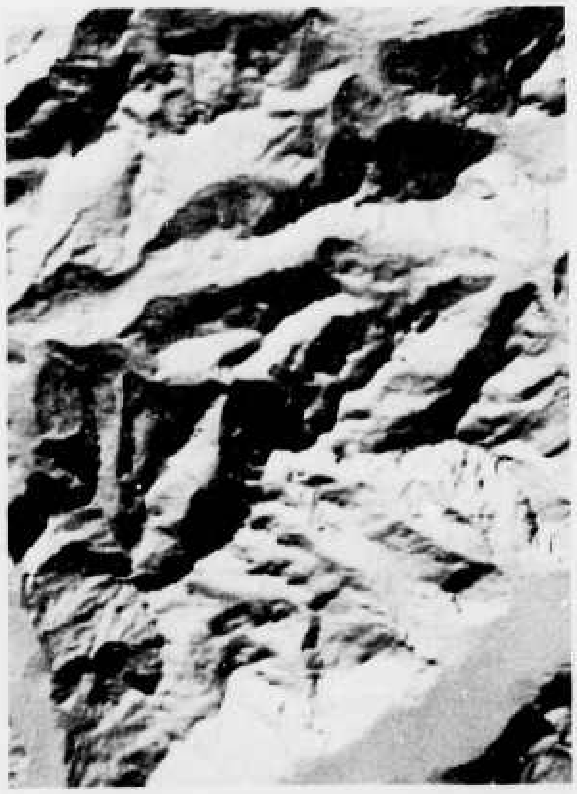
This work was described in detail by Grimson and Marr in the April, 1979 Proceedings. Grimson is working on the difficult problem of interpolating the depth for areas between zero-crossing contours.

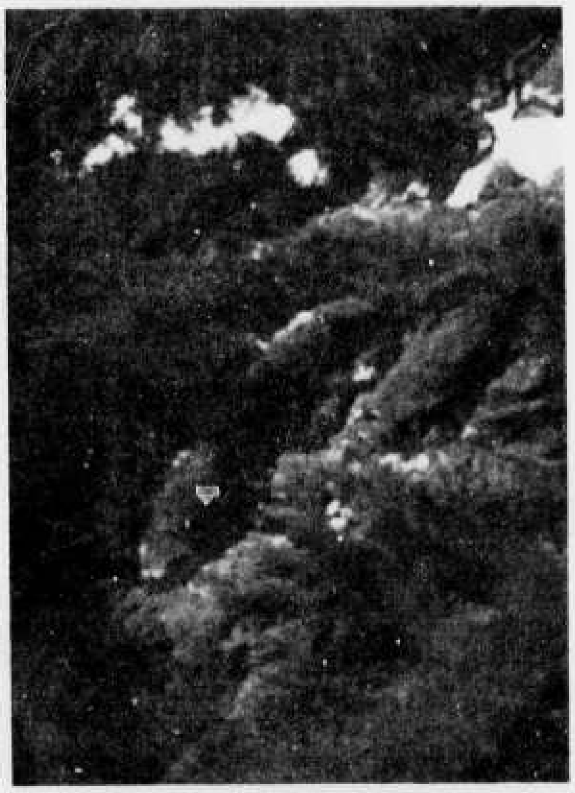
REFERENCES

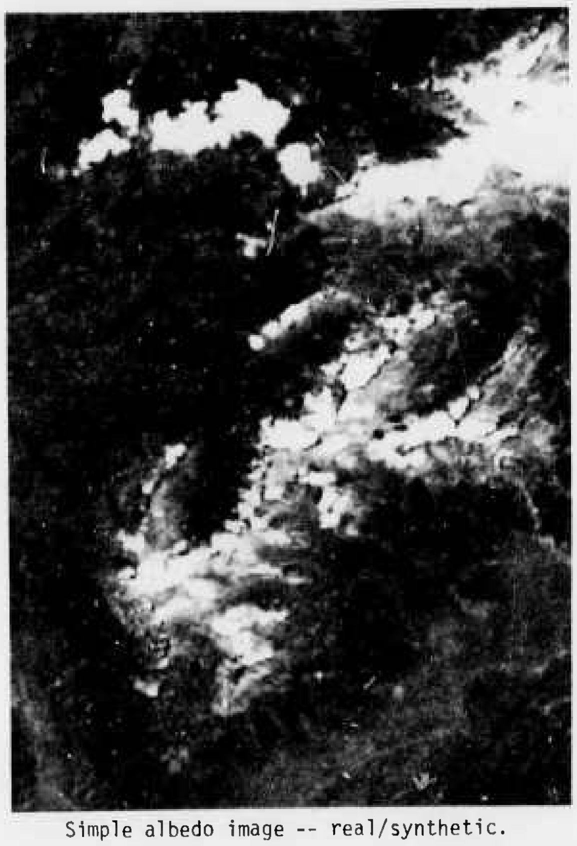
For an extended discussion of MIT work on Image Understanding, see Volume 2 of *Artificial Intelligence: an MIT Perspective*, edited by Patrick H. Winston and Richard H. Brown, MIT Press, Cambridge, Massachusetts, 1979.

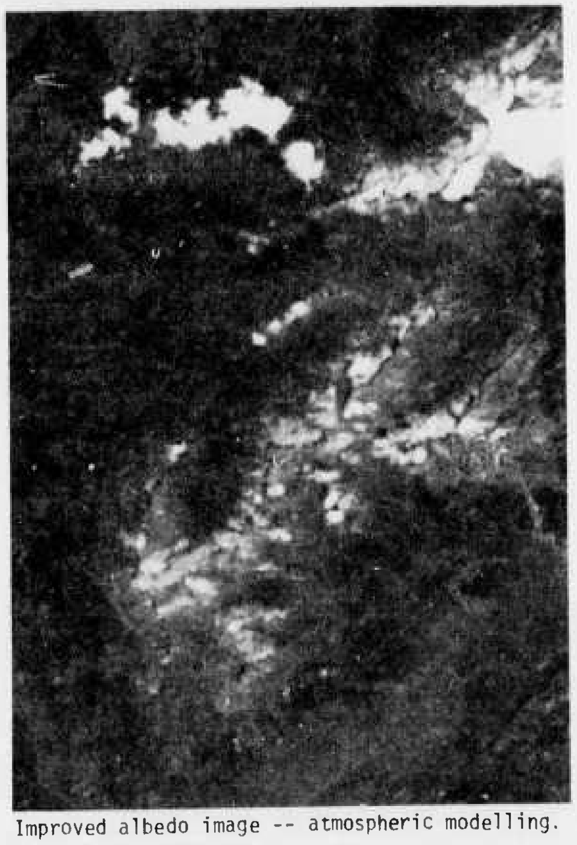
W. E. L. Grimson and David Marr, "A Computer Implementation of a Theory for Human Stereo Vision, November, 1978 Proceedings: Image Understanding Workshop.

Berthoid K. P. Horn, "Hill-Shading and the Reflectance Map" April, 1979 Proceedings: image Understanding Workshop.


Synthetic image with cast shadows.


Destriped, rectified LANDSAT image.


Simple albedo image -- real/synthetic.


Improved albedo image -- atmospheric modelling.

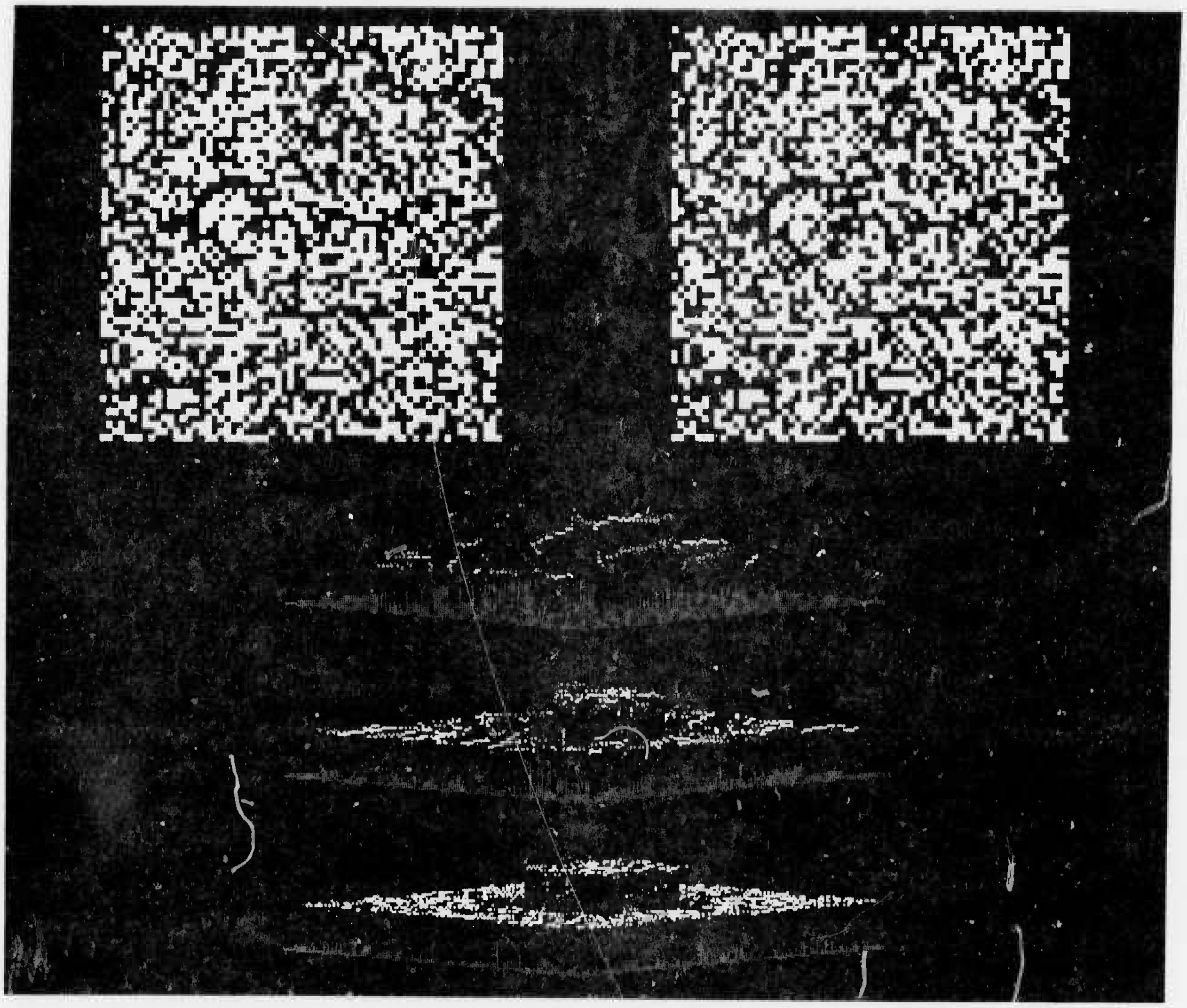


Figure 4: Random-dot stereo pairs provide a hard challenge for stereo algorithms. The three elevation maps show the progress of the depth computation starting with the course match using the most blurred image pair.

Berthold K. P. Horn and Brett L. Bachman, "Using Synthetic Images to Register Real Images with Surface Models," October, 1977 Proceedings: Image Understanding Workshop. Also AIM-437, The Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1977. Also in *CACM*, November, 1978.

Berthold K. P. Horn and Robert W. Sjoberg, "Calculating the Reflectance Map," November, 1978 Proceedings: Image Understanding Workshop. Also AIM-495, The Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1978. Also in *Applied Optics*, June, 1979.

Berthold K. P. Horn and Robert J. Woodham, "Destriping Satellite Images," May, 1978 Proceedings: Image Understanding Workshop. Also AIM-467, The Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, Massachusetts 1978. Also to appear in *CGIP*.

IMAGE UNDERSTANDING USING OVERLAYS
Project Status Report
1 April-30 September 1979

Computer Vision Laboratory
Computer Science Center
University of Maryland
College Park, MD 20742

Azriel Rosenfeld, Principal Investigator

ABSTRACT

Current activities on the project are reviewed under the following headings:
1) Image modelling and preprocessing
2) Edge detection and linking
3) Segmentation
4) Texture analysis
5) Shape analysis and matching
6) Hierarchical region representation

Some other related work at the Computer Vision Laboratory is also briefly mentioned.

1. INTRODUCTION

This project is concerned with the study of advanced techniques for the analysis of reconnaissance imagery. It is being conducted under Contract DAAG-53-76C-0138 (DARPA Order 3206), monitored by the U.S. Army Night Vision Laboratory, Ft. Belvoir, VA (Dr. George Jones). The Westinghouse Systems Development Division, as a subcontractor, is investigating hardware implementation of the techniques being developed by Maryland, particularly in the area of relaxation; their efforts are reviewed in separate quarterly reports.

The current phase of the project is concerned with the development and application of advanced techniques for image processing, feature detection, segmentation, texture and shape analysis, and region representation. These aspects are reviewed in the following sections. This report deals primarily with the work done during the past six months; activities during earlier periods were reviewed in previous reports [1-4]. Some of the topics are discussed only briefly, since they are treated in greater detail in other papers in these Proceedings.

2. IMAGE MODELLING AND PREPROCESSING

Image models

Many types of statistical models for images have been developed; they include random field and time series models, as well as models for region shapes in terms of border or run sequences. An NSF/ONR sponsored Workshop on Image Modelling was organized and held in August 1979; over 25 papers were presented on various aspects of image modelling. Under an AFOSR grant, research is being conducted on a class of "mosaic" image models based on random geometric processes, as well as on the problem of neighborhood selection in time series and random field models.

The last status report briefly reviewed some of the basic types of mosaic models and their properties, and also reported on some preliminary studies in fitting such models to real textures. During the current reporting period, further studies on mosaic model fitting were conducted [5]. Six cell structure models were used: the checkerboard, hexagonal, triangular, Poisson line, occupancy, and Delaunay models. Some experiments were also carried out using coverage or "bombing" models, but the fits obtained were much poorer than those of the cell structure models.

The models were fitted to samples of seven textures: three geological terrain types on a LANDSAT image (Pennsylvanian sandstone and shale, Mississippian limestone and shale, and Lower Pennsylvanian shale) and four textures from Brodatz's album (grass, raffia, sand, and wool). These textures were chosen because they have been used in earlier texture classification experiments by various investigators. "Patches", or primitive elements, were extracted from each texture using an edge-based method (see Section 5). Small patches (having less than 13 pixels) were discarded to reduce noise.

For each of the models, the expected patch area and patch perimeter can be derived as functions of a parameter representing the "density" of the mosaic. By adjusting this parameter, we can fit the expected area to the observed area for a given texture sample. The difference between the expected and observed perimeters can then be used as a measure of how well the model fits the texture. Conversely, we can adjust the parameter to make the expected and observed perimeters agree, and then use the difference between the expected and observed areas as a measure of the fit.

Table 1 shows the fits obtained in this way for the six models and for several samples of each of the seven textures. It is seen that:

a) For most of the textures, there is a consistently best fitting model
b) This best model is usually the same whether the area or perimeter is used for fitting the model
c) The goodness of fit of the various models to a given texture usually spans a wide range; the best model often fits substantially better than the next best one.

The mosaic models used in these experiments were chosen because they are mathematically tractable (so that quantities such as the expected area and perimeter can be computed for them), not because we consider them to be realistic models for natural textures. Nevertheless, the results of our model fitting experiments indicate that some of these models provide good predictions of the properties of real textures. Thus mosaic models seem to be a useful addition to the repertoire of mathematical tools for image and texture modelling.

Preprocessing techniques

During the previous reporting periods, comparative tests were conducted on a variety of noise cleaning techniques, primarily based on iterated local operations, and a number of new techniques of this type were developed. Some further variations are currently under study. All of these methods assume that the ideal image is piecewise constant; the general idea is to average each pixel with a subset of its neighbors, namely those which appear to belong to the same region as the pixel. When such methods are applied to a noisy image, the peaks on its histogram tend to become much sharper, implying that the gray level variability in the regions has been reduced. A two-level method has also been developed in which link strengths are computed between each pixel and its neighbors, and smoothing is done by local averaging in which the weight given to each neighbor depends on its link strength; both the local averaging and the link strength computation are iterated. When this method is used, the image histogram tends to become spike-like.

A class of "probability transforms" of images has been defined [6]. The basic idea is as follows: One or more local properties (e.g., gray level, gradient magnitude, etc.) are measured for each pixel. We take the frequencies of occurrence of the property values as estimates of their probabilities, and the joint or conditional frequencies of occurrence of pairs of values as estimates of the joint or conditional probabilities. For each pixel, we can display its probability with respect to any given property, or its joint or conditional probability with respect to any given pair of properties, as a gray level. Some of the results seem to be useful in enhancing subtle properties of the original image, as Figure 1 illustrates. Haralick, who first defined the joint gray level probability transform, suggests that it may be useful in texture analysis. These transforms tend to be quite sensitive to noise; but as mentioned in an earlier report, they may also be useful in connection with noise cleaning.

3. EDGE DETECTION AND LINKING

Cultural features such as roads and buildings on aerial photographs usually have relatively sharp edges that are piecewise straight or smooth and that occur in antiparallel pairs, i.e., with their dark sides or light sides facing one another. Thus a useful approach to the extraction of such features is to detect edge segments and link them into groups based on relationships of collinearity, good continuation, and (anti-) parallelness. Such an approach will be described in a separate paper in these Proceedings [7]. In this section we briefly discuss some of the motivations that underlie this approach.

Ideally, one might want to use knowledge about the properties of the features at every stage of the extraction process. However, at the initial stage of this process, much of this knowledge is simply not useful. When examining individual pixels and their neighborhoods, it is not usually possible to decide whether a pixel is on (the edge of) a road or building, unless roads and buildings have distinctive gray levels (or colors) or contrasts. Similarly, the relationships among individual pixels do not provide much information about whether they are parts of roads or buildings, since these features can have a variety of sizes and shapes, and even if we assume that a pixel belongs to a feature, we do not know to what part of the feature it belongs. Thus at the pixel level it is more appropriate to use "local" knowledge about primitive entities such as edges, lines, corners, etc., as they occur in the features of interest, rather than attempting to use knowledge about the features themselves. The result of processing at this level might thus be a set of edge segments, line segments, etc.

A somewhat higher level of knowledge can be used in defining groups of these segments. Here one can emply "semilocal" information about primitive shape properties such as straightness and parallelism; but it is still difficult to use global shape information, or information as to how features interact with one another, at the segment level, since the positions of the segments within the features are not generally known. Thus an appropriate goal at this stage might be the extraction of "subfeatures" or "feature segments", e.g. by linking collinear groups of line segments, or groups of edge segments that form an antiparallel strip. Further grouping of the feature segments into features can then be carried out; at this stage, it should be possible to make effective use of global information.

An approach to cultural feature extraction, based on these stages, is described in [7], where we discuss the methods used at each stage and give preliminary examples of results obtained at the first two stages.

4. SEGMENTATION

Thresholding

If we use a region-based, rather than edge-based, approach to feature extraction, then a common initial step is to classify the image pixels based on their gray levels (or colors)--i.e., to threshold the image. Many methods have been suggested for choosing the threshold. One possibility is to use one-dimensional versions of standard clustering algorithms. For example [8], the well-known ISODATA algorithm, as applied to the thresholding problem, might operate as follows: (a) start with an arbitrary threshold; (b) compute the mean gray levels of the resulting two sets of pixels; (c) pick a new threshold midway between these means; repeat steps (b-c). It can be shown that this process always converges, and that it yields good thresholds for images that do contain two populations of pixels. More generally, the process can be used to classify the pixels into any specified number of classes, i.e., to requantize the image into a given number of gray levels.

In the previous status report, an application of relaxation to thresholding was described. Initial "light" and "dark" probabilities are assigned to each pixel based on its gray level, and these probabilities are then iteratively adjusted based on the probabilities at neighboring pixels, with light reinforcing light and dark dark. Within a few iterations, the histogram of probabilities (which can be displayed as gray levels) begins to turn into two spikes at the ends of the gray scale. This method assumes that the two desired classes are 'light" and "dark"; we now describe an alternative approach in which any desired gray level classes can be used. Here the initial probabilities are determined by fitting a sum of standard distributions, e.g. Gaussians, to the image's histogram, where each Gaussian defines a gray level class. Bayes' theorem can then be used to determine the probability that a given gray level belongs to each of the classes, and these initial probabilities are then iteratively adjusted as before. Figure 2 shows an example of the application of this method to a FLIR image of a tank; here there are two classes, and the results are displayed by representing the class probabilities as positions along the grayscale. Note that even the initial probabilities, when displayed in this way, give rise to two peaks at the ends of the grayscale, with very little between them; thus very little remains to be done on the subsequent iterations.

Blob detection

An approach to blob detection using relaxation was described in the previous status report [9]; it used two interacting relaxation processes, one dealing with light and dark probabilities, the other with edge and no-edge probabilities, and gave better results than could be obtained using either process alone. A more detailed description of this work appears in [10]. Another possible source of information relevant to blob detection might be "interior" and "exterior" probabilities, initially set to .5 except adjacent to borders, where the interior probability is higher on the concave side of a curved border segment, and the exterior probability is higher on the convex side. An interior/exterior relaxation process was implemented in [10] in the binary case, where the object/background borders are known; under this process, the interior probabilities go to 1 inside the object and the exterior probabilities go to 1 outside it, even if initially these probabilities have the wrong relative sizes due to the presence of concavities in the border. Generalizations of the interior/exterior process to unsegmented images are currently under investigation. One possibility is to compute a gradient direction at each point, and to estimate the curvature at a point by the rate of change of this direction; if desired, the resulting interior/exterior probabilities can be "attenuated" (i.e., moved closer to .5) if the gradient magnitude is small. An even simpler possibility is to use the numbers of neighbors of a point that are lighter (darker) than the point as a basis for defining "curvature" at the point.

In the previous status report it was suggested that multiple-resolution ("pyramid") array representations could provide a basis for introducing simple types of size and shape information into the segmentation process. For example, blobs become local features ("spots") at some level of the pyramid; thus if a spot is detected, we can adjust the parameters of the full-resolution segmentation processes in that vicinity so as to favor extraction of the spot. A simple version of this idea has been implemented [11], and is described in a separate paper in these Proceedings. Basically, when a spot is detected, a local threshold is chosen midway between the average gray levels of the center and surround regions of the spot detector; this threshold extracts the spot very well. Planned extensions of this approach will employ other types of detectors, and will allow the results to influence the segmentation process in additional ways.

5. TEXTURE ANALYSIS

Texture plays an important role in the classification of terrain and land use types on aerial photographs and remote sensor imagery, particularly if multispectral information is not available.

Several texture analysis efforts are currently in progress; one of them, involving second-order gray level statistics computed at points defined by characteristic local property values, was described in the previous status report. Some of the current work is described in a separate paper in these Proceedings [12]; it deals with cooperative computational methods in texture analysis, in connection with the adjustment of texture feature values as well as the extraction of texture primitives. The main lines of investigation are briefly summarized in the following paragraphs.

In the absence of color information, sets of local property values can sometimes be used as features for pixel classification. However, these values tend to be quite variable; for example, even in a "busy" region, local busyness measures do not have uniformly high values. Iterative smoothing methods, or relaxation methods, can be applied to the values to make them more consistent. Similar remarks apply to textural properties computed for windows of an image; one wants to use small windows in order to make it less likely that a window overlaps two differently textured regions, but for small windows the property values are quite variable. Here again, iterative smoothing methods can be used to reduce the variability, resulting in reliable classification even for small windows. Further work along these lines is planned, involving comparisons among windows of different sizes.

Several simple methods of extracting texture primitives from an image have been investigated, including thresholding at a percentile, adaptive requantization (converting the image's histogram into a small set of spikes), and the SUPERSLICE segmentation algorithm. The resulting primitives are generally not "clean", but statistics computed from them (area, perimeter, elongatedness, etc., as well as second-order statistics computed for neighboring pairs of primitives) are nevertheless useful for texture classification. Much "cleaner" primitives can be obtained using an edge-based approach [13], in which primitives are detected as clusters of antiparallel edge pairs. This approach can be modelled in several ways, e.g. using "dipoles" of varying length and orientation, and detecting clusters of dipole responses; these responses can be based on the gradient magnitudes and directions at the two ends of each dipole, rather than requiring decisions to be made as to whether or not edges are present. Another approach is to use simple spot (or streak) detectors to detect the positions of texture primitives; the primitives can then be extracted by local segmentation techniques such as that described in the preceding section.

6. SHAPE ANALYSIS AND MATCHING

A relaxation-based approach to shape analysis and matching was described in the previous status report, and has now been documented in greater detail in two technical reports [14,15]. The first phase of this work [14] dealt with single, isolated shapes, and addressed the problem of matching a shape to a model in cases where the shape cannot be unambiguously segmented. The approach taken was to represent the ambiguous segmentation as a graph in which the nodes are the segments, and the arcs link pairs of segments that are consecutive along the border. The nodes are then probabilistically classified as being various parts referred to by the model--e.g., nose, wings, and tail, in an airplane. Relaxation is used to adjust these probabilities; this greatly reduces the ambiguity of the graph, and allows matches to the model (i.e., cycles consisting of the proper sequences of parts) to be found quickly.

In [15], this approach is extended to handle sets of touching shapes. Here the (ambiguous) segments are linked based on proximity and continuation relationships; in other words, the relationship of consecutivity is now also treated as ambiguous (e.g., when two shapes touch, it may not be obvious how to pair up the border segments at the point of contact). Paths in the resulting graph are probabilistically classified as being parts of the shape, and relaxation is applied to increase the probabilities of classifications that support one another In this case too, the relaxation process results in a high degree of disambiguation. An example, involving four touching airplane shapes, is shown in Figure 3.

Further extensions to the approach are needed to handle shapes with missing parts. An extension to hierarchical shape models (e.g., decomposing the nose, wings, and tail into subparts) would also be desirable.

Methods of shape segmentation are also under study. The segmentations used in the studies described above are based on detection of curvature extrema on the border; thus these segmentations are based on relatively local evidence. A more global approach to shape segmentation involves finding arcs for which a given function has a locally extremal value. For example, if the function is arc length divided by chord length, it should have a maximum when the chord just cuts across the base of a protrusion or intrusion. A number of such functions are being investigated; a report on their properties and their usefulness for shape segmentation is in preparation.

7. HIERARCHICAL REGION REPRESENTATION

Extensive studies have been conducted on the use of quadtree structures as representations of binary images. A separate paper reviewing this work appears elsewhere in these Proceedings [16]. Ten technical reports [17-26] have been issued on this work, in addition to the two reports issued during the last reporting period.

The quadtree representation of a 2^n-by-2^n binary array is constructed as follows: If the array consists entirely of 1's or 0's, its tree consists of a single "black" or "white" node, respectively. Otherwise, we subdivide the array into quadrants, and represent it by a "gray" node having four descendants corresponding to the quadrants. The process is then repeated for each quadrant--i.e., if it consists entirely of 1's or 0's, it is represented by a black or white node, and if not, it is represented by a gray node having four descendants corresponding to its subquadrants. This process is iterated until no further subdivisions into quadrants are necessary. The result is a tree whose root node corresponds to the entire array and whose leaf nodes correspond to blocks consisting entirely of 1's or 0's. Each leaf node is black or white, and each gray (non-leaf) node has four descendants.

This representation can be generalized to non-binary images; here a quadrant is subdivided unless its pixels all have the same value. However, the representation is economical only for images composed of large regions of constant value; if many gray levels are possible, this is unlikely.

Efficient algorithms have been designed for converting between quadtrees and other representations, and for computing various properties of an image directly from its quadtree. These algorithms typically operate by traversing the tree; their execution time depends on the number of tree nodes, and not on the sizes of the blocks that these nodes represent, so that for compact quadtrees they are very fast. An overview of these algorithms is given in a separate paper in these Proceedins [16]. They include: performing Boolean operations on binary images represented by quadtrees [23]; computing moments [23]; computing perimeter [17]; labelling connected components [18]; computing the genus [22]; computing a form of city block or chessboard distance from each black node to the nearest white node [24,25]; and determining a quadtree "medial axis transformation" based on this distance [26]. Other algorithms deal with efficient bottom-up quadtree construction [20], and with conversion between run length and quadtree representations [29,21], as well as with conversion between quadtrees and border codes (reported previously). Experimental studies are in progress on the use of quadtrees to define approximations to binary images, and on the accuracy of estimating shape properties from these approximations; this work will be the subject of a subsequent report.

8. OTHER PROJECTS

We conclude by briefly mentioning some of the other projects currently being conducted at the Computer Vision Laboratory.

a) Under NSF Grant MCS-76-23763, the theory and performance evaluation of relaxation processes for iterative probabilistic classification is being studied. Techniques for image segmentation based on multiple-resolution operations are also under investigation, and a combination of the two approaches is planned. This work also has obvious applications to the image analysis tasks with which the present project is concerned.

b) Under NSF grant MCS-77-18719, a transportable Fortran-based image processing software system has been designed. This work supplements the ongoing development of utility software within the Laboratory; see [27] for documentation of a collection of this software.

c) Under AFOSR grant AFOSR-77-3271, in addition to research on image modeling, extensive studies are being conducted on the theory of cellular processors, both array- and graph-structured, having either fixed or reconfigurable structures. As an outgrowth of this work, a cellular processor consisting of several hundred microprocessors is being designed. The proposed design is described in a separate paper in these Proceedings [28].

References

1. Semi-annual report, 1 April-30 September 1978.

2. Semi-annual report, 1 October 1978-30 March 1979.

3. Project Status report, Proceedings, Image Understanding Workshop, November 1978, 20-27.

4. Project Status report, Proceedings, Image Understanding Workshop, April 1979, 14-24.

5. N. Ahuja and A. Rosenfeld, Fitting mosaic models to textures, Computer Science TR-789, University of Maryland, College Park, MD, July 1979.

6. A. Scher and A. Rosenfeld, "Probability transforms" of digital pictures, Computer Science TR-758, University of Maryland, College Park, MD, April 1979.

7. A. Rosenfeld, Levels of representation in cultural feature extraction, Proceedings, Image Understanding Workshop, November 1979.

8. F. R. D. Velasco, Thresholding using the ISODATA clustering algorithm, Computer Science TR-751, University of Maryland, College Park, MD, March 1979.

9. A. Rosenfeld, A. Danker, and C. R. Dyer, Blob extraction by relaxation, Proceedings, Image Understanding Workshop, April 1979, 61-65.

10. A. Danker, Blob detection by relaxation, Computer Science TR-795, University of Maryland, College Park, MD, July 1979.

11. M. O. Shneier, Using pyramids to define local thresholds for blob detection, Computer Science TR-808, University of Maryland, College Park, MD, September 1979; also in Proceedings, Image Understanding Workshop, November 1979.

12. A. Rosenfeld, Cooperative computation in texture analysis, Proceedings, Image Understanding Workshop, November 1979.

13. T. H. Hong, C. R. Dyer, and A. Rosenfeld, Texture primitive extraction using an edge-based approach, Computer Science TR-763, University of Maryland, College Park, MD, May 1979.

14. W. S. Rutkowski, S. Peleg, and A. Rosenfeld, Shape segmentation using relaxation, Computer Science TR-762, University of Maryland, College Park, MD, May 1979.

15. W. S. Rutkowski, Shape segmentation using relaxation, 2, Computer Science TR-793, University of Maryland, College Park, MD, July 1979.

16. H. Samet and A. Rosenfeld, Quadtree structures for region processing, Proceedings, Image Understanding Workshop, November 1979.

17. H. Samet, Computing perimeters of images represented by quadtrees, Computer Science TR-755, University of Maryland, College Park, MD, April 1979.

18. H. Samet, Connected component labeling using quadtrees, Computer Science TR-756, University of Maryland, College Park, MD, April 1979.

19. H. Samet, Region representation: raster-to-quadtree conversion, Computer Science TR-766, University of Maryland, College Park, MD, May 19[illegible].

20. H. Samet, Region representation: quadtrees from binary arrays, Computer Science TR-767, University of Maryland, College Park, MD, May 1979.

21. H. Samet, Region representation: quadtree-to-raster conversion, Computer Science TR-768, University of Maryland, College Park, MD, June 1979.

22. C. R. Dyer, Computing the Euler number of an image from its quadtree, Computer Science TR-769, University of Maryland, College Park, MD, May 1979.

23. M. Shneier, Linear time calculations of geometric properties using quadtrees, Computer Science TR-770, University of Maryland, College Park, MD, May 1979.

24. H. Samet, A distance transform for images represented by quadtrees, Computer Science TR-780, University of Maryland, College Park, MD, July 1979.

25. M. Shneier, A path-length distance transform for quadtrees, Computer Science TR-794, University of Maryland, College Park, MD, July 1979.

26. H. Samet, A quadtree medial axis transform, Computer Science TR-803, University of Maryland, College Park, MD, August 1979.

27. R. L. Kirby, R. Smith, P. A. Dondes, and F. Blonder, Interfaces, subroutines, and programs for the Crinnell CMR-27 display processor, on a PDP-11/45 with the UNIX operating system, Computer Science TR-810, University of Maryland, College Park, MD, October 1979.

28. C. J. Rieger III and J. Bane, ZMOB: A mob of 256 Z80A-based microcomputers, Proceedings, Image Understanding Workshop, November 1979.

TEXTURE	SAMPLE	MODEL C	H	T	P	O	D
WOOL	1	.08	.06	.15	.17	.08	.23
		.04	.03	.08	.08	.04	.14
	2	.21	.23	.11	.53	.06	.00
		.09	.10	.05	.19	.03	.00
	3	.09	.08	.01	.39	.06	.09
		.04	.04	.01	.15	.03	.05
RAFFIA	1	.17	.33	.33	.49	.42	.20
		.08	.22	.13	.18	.32	.09
	2	.14	.32	.26	.46	.41	.14
		.07	.21	.11	.17	.30	.06
	3	.32	.17	.57	.68	.28	.37
		.13	.10	.19	.23	.18	.15
SAND	1	.75	.01	.96	1.23	.13	.77
		.24	.00	.29	.33	.07	.25
	2	.83	.04	1.08	1.33	.10	.88
		.26	.02	.31	.34	.05	.27
	3	.62	.10	.92	1.06	.23	.73
		.21	.05	.28	.30	.13	.24
GRASS	1	.89	.12	1.08	1.40	.03	.88
		.27	.06	.31	.35	.01	.27
	2	1.14	.36	1.26	1.72	.17	1.04
		.32	.14	.33	.39	.08	.30
	3	.74	.14	.71	1.21	.02	.54
		.24	.06	.23	.33	.01	.19

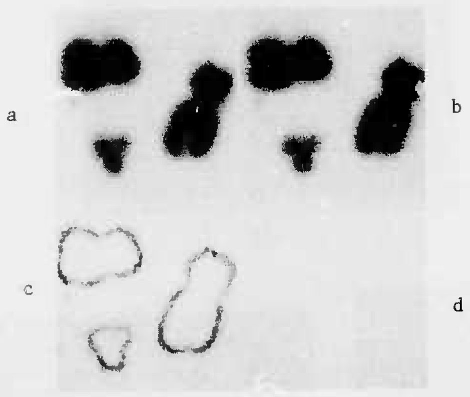
TEXTURE	SAMPLE	MODEL C	H	T	P	O	D
MISSISSIPPIAN	1	.03	.19	.05	.31	.03	.14
		.01	.08	.03	.13	.01	.08
	2	.02	.13	.10	.24	.02	.19
		.01	.06	.05	.10	.01	.11
	3	.18	.37	.09	.51	.18	.01
		.08	.14	.04	.19	.08	.01
	4	.44	.33	.37	.84	.16	.23
		.17	.13	.15	.26	.07	.10
PENNSYLVANIAN	1	.83	.15	1.11	1.33	.00	.91
		.26	.07	.31	.35	.00	.28
	2	.54	.04	.57	.96	.17	.41
		.19	.02	.20	.29	.10	.16
	3	.45	.09	.48	.85	.22	.33
		.17	.05	.18	.26	.13	.13
	4	1.19	.39	1.32	1.79	.20	1.09
		.32	.15	.34	.40	.09	.31
LOWER PENNSYLVANIAN	1	.30	.19	.16	.66	.03	.04
		.12	.08	.07	.22	.01	.02
	2	.24	.22	.14	.57	.06	.03
		.10	.10	.06	.20	.03	.01
	3	.16	.34	.07	.48	.16	.03
		.07	.14	.03	.18	.07	.02
	4	.21	.20	.12	.54	.04	.01
		.09	.09	.05	.19	.02	.00

Table 1. Errors in fitting six cell structure models to samples of seven textures.

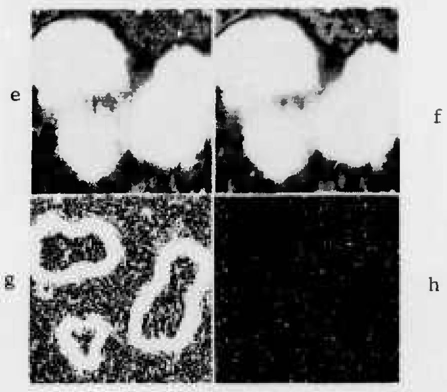
C = Checkerboard, H = hexagonal, T = triangular

P = Poisson line, O = occupancy, D = Delaunay

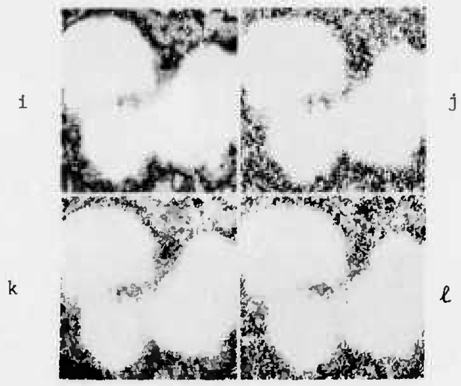
For each texture, the odd-numbered rows show errors in perimeter when the predicted and observed cases are matched; the even-numbered rows show area errors when perimeters are matched. Each pair of rows represents a different texture sample; there are three samples each of wool, raffia, sand, and grass, and four samples each of the three geological terrain textures.



a) Original picture (three chromosomes)
b) 3x3 local average
c) Roberts gradient magnitude
d) Laplacian magnitude

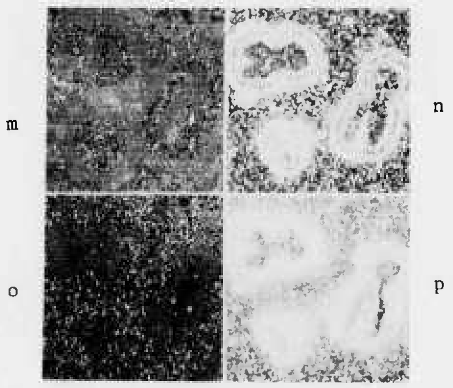


(e-h) First-order probability transforms of (a-d): The probability of the value at each point is rescaled and displayed as a gray level.



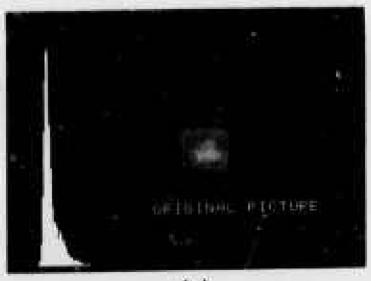
(i-ℓ) Joint probability transforms for gray level and four properties. The joint probability of the gray level and property value at each point is rescaled and displayed as a gray level. The properties are

i) Laplacian magnitude
j) Roberts gradient magnitude
k) 3x3 local average gray level
ℓ) gray level of right-hand neighbor

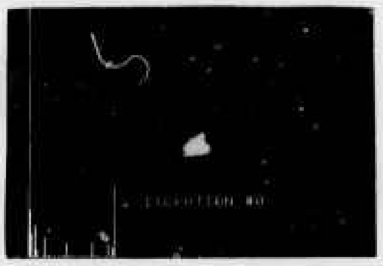


(m-p) Conditional probability transforms for the same four properties. The conditional probability of the property value given the gray level at each point is rescaled and displayed as a gray level.

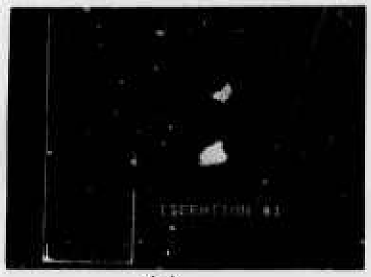
Figure 1. A set of "probability transforms".



(a)



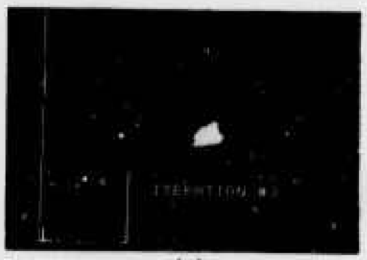
(b)



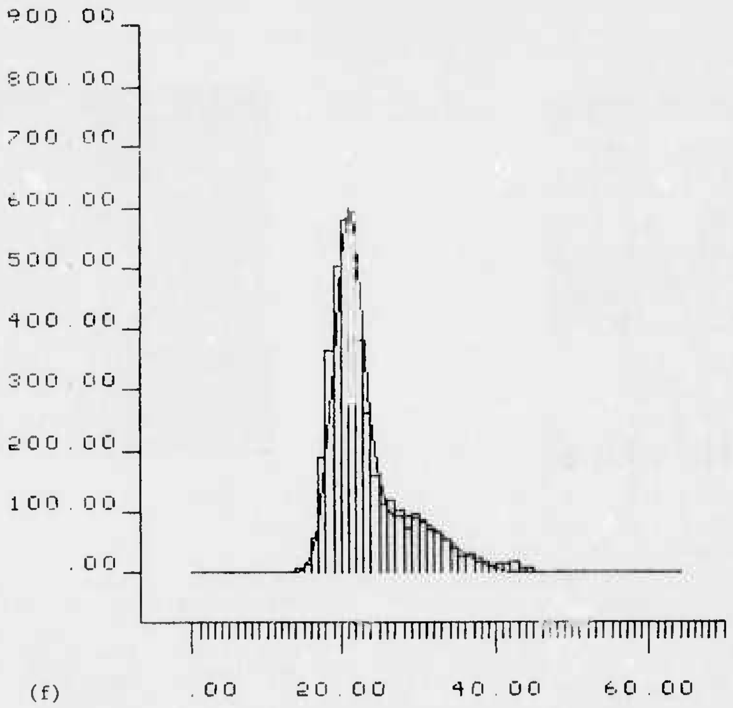
(c)

ITERATION #2

(d)



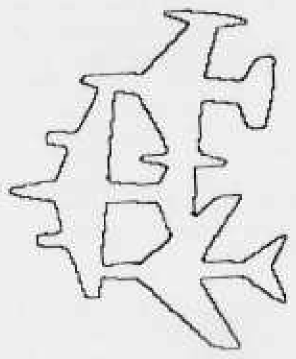
(e)



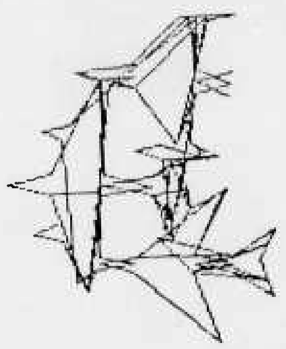
(f)

(a) The original picture (a tank on a FLIR image) and its histogram.
(b) Result of fitting a sum of two Gaussians to the histogram, computing the probability that each gray level belongs to the two classes, and displaying the probability of membership in the lighter class (rescaled) as a gray level. The resulting histogram (of rescaled probabilities) now has two predominant spikes at the end of the scale, even before relaxation is applied.
(c-e) Results of three iterations of relaxation applied to (b). The first iteration eliminates nearly all of the intermediate values; there is little change after this iteration.
(f) Plot of the histogram, showing the two-Gaussian fit.

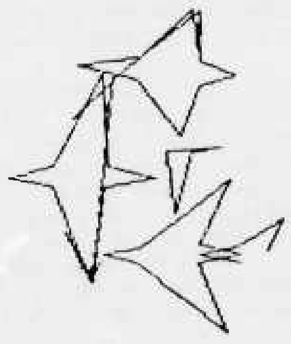
Figure 2. Thresholding by Gaussian fitting and relaxation.



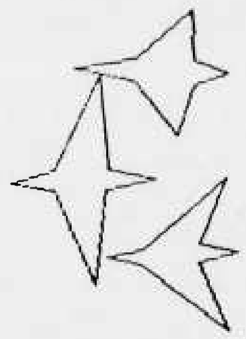
a.



b.
Iteration 0



c.
Iteration 1



d.
Iteration 2

(a) Input: three touching airplane shapes

(b) Two-piece approximating polygons for each of 53 initial segments

(c-d) Two-piece approximating polygons for the segments remaining after iterations 1 and 2 of relaxation. The second iteration has separated the input into three disjoint planes and has produced a unique segmentation of each plane into four parts.

Figure 3. Segmentation of touching shapes by relaxation.

IMAGE UNDERSTANDING RESEARCH AT CMU: A Progress Report

Raj Reddy and John R. Kender
Department of Computer Science
Carnegie-Mellon University
Pittsburgh, PA 15213

INTRODUCTION

Our research objective continues to be the development of image understanding techniques within the framework of an integrated demonstration system. This is a two-pronged effort: the analysis and codification of many available sources of visual knowledge, and the synthesis and coding of a user-friendly environment. For our latest six-month segment we report progress in basic algorithm research and enhancement, in addition to the results of our major concentration on system development, both hardware and software.

Our three task areas remain the same. We continue to investigate the application of diverse knowledge-based techniques in the understanding of two-dimensional satellite images of Washington, D.C., and of three-dimensional color scens of downtown Pittsburgh. As before, the stress in this area is on cost-effectiveness. As reported below, much of the creation of new low-level and high-level interpretation techniques is done with an eye towards their eventual integration into a working signal-to-symbol system.

The second task area has produced a very tangible result. Our concern with the development and verification of suitable computer architectures for computer vision has resulted in a breadboard demonstration of the feasibility of a VLSI chip. Our other similar efforts are motivated by the necessity for fast and cheap image computations, in research as well as in application domains.

Our third area concerns the development of interactive aids for image processing applications. We have directed our work towards many of the software issues that such systems imply, particularly those involving the management of the data base.

What follows summarizes briefly our progress this past half-year.

SYSTEMS

We are currently adapting much of the software from our multi-processor Image Understanding System to a DEC VAX running UNIX. We are augmenting this machine with a Grinnell display and an interface to our existing special-purpose median-filter TI board. (The present hardware configuration is being retained as a specialized display and database computer, addressable over our in-house network.) This is a sizable tool-building effort that includes the refitting of picture paging and displaying algorithms. We expect it will be well repaid in research efficiency, principally due to the virtues of shared code. In this last regard, we will continue to program in C until a reliable Ada compiler is available.

TASKS

Research on more specific, task-related issues includes the following. The concentration of effort in the aerial imagery task has been on the questions of symbolic representation of data. This work is reported in a companion paper (McKeown, 1979). The basic search strategy in the downtown Pittsburgh task has been redesigned, implemented, analyzed, and (favorably) compared with the original. Several other aspects of the original system are better understood and controlled; this work is found in a second companion paper (Smith, 1979). Lastly, more theoretic results have been derived from the shape-from-texture paradigm (Kender, 1979a). We intend to incorporate these new algorithms and representations into the three-dimensional Pittsburgh task. Some of them are outlined below.

In representing local shape, it appears that the Gaussian sphere is a powerful tool. Very strong analogies are shown to exist among the families of constraint curves that are generated by the separate methods that derive shape from shading, from texture, and from occluding contours. The analogies are not only strong, they are exceedingly simple: in this representation, they are latitude or longitude lines on the (appropriately tilted) sphere. An elegant similarity also is found between closely-illuminated objects and objects viewed under perspective. Further, use of this representation in lieu of the gradient space allows extension of the method of shape-from-shading to handle cases where illumination is in front of the camera: the gradient space is only half the necessary space.

In calculating local shape, several new methods neatly fall out from the application of the paradigm. Some of these are exact; others are approximate, in the sense of function approximation. Among the exact methods are those based on the assumption of equal lengths or spacings. A direct consequence of these (very simple) relations is the strong suggestion that inverse distance is a highly appropriate measure in dealing with three-dimensional scenes. Another application of the paradigm to gravity-sensitive scene assumptions (horizontal, vertical), shows how such information can be easily integrated into the representational scheme. Still other applications yield further results (Kender, 1979b).

In general, our hope is that all our research (and the research of others) can be fruitfully harmonized in our application vehicle. We perceive that this necessitates the development of an ample collection of low-level visual experts; shape-from-texture is one such. Each specialist cooperatively contributes to an intermediate, symbolic data structure on which the higher-level knowledge-based search and deduction schemes depend.

A computational organization like this articulates a theory of image abstraction. One graphic equivalent is the generation of a series of map overlays from an aerial view. In this framework, the aerial imagery task is the intelligent goal-driven selection and integration of many signal-driven representations; the goal is the overlay.

ARCHITECTURES

Our special-purpose image processing hardware progresses steadily. The Texas Instruments VLSI board has arrived and is being interfaced to our VAX. TI is building another board; this one will perform a three-by-three convolution at video rates. The convolution coefficients are programmable (Eversole, 1979). Meanwhile, Control Data is constructing the SPARC ultra-high speech signal processing computer (Allen, 1979). It is expected to arrive in January of 1980. Its arrival will undoubtably spark another round of intensive software implementation, debugging, and improvement, similar to the one just reported concerning our VAX.

REFERENCES

G.R. Allen, P. G. Juetten, "SPARC--Symbolic Processing Algorithm Research Computer," Proceedings of the ARPA Image Understanding Workshop, Science Applications Inc., Apr., 1979.

W. L. Eversole, D. J. Mayer, F. B. Frazee, T. F. Cheek, Jr., "Investigation of Advance Real-Time Image Understanding System," Proceedings of the ARPA Image Understanding Workshop, Science Applications Inc., Apr., 1979.

J. R. Kender, "Shape from Texture: A Computational Paradigm," Proceedings of the ARPA Image Understanding Workshop, Science Applications Inc., Apr., 1979.

J. R. Kender, "Shape from Texture," Ph.D. Thesis, Computer Science Dept., Carnegie-Mellon Univ., 1979 (in preparation).

D. M. McKeown, "Representations for Image Data Bases," in this volume.

D. R. Smith, "Search Strategies for the ARGOS Image Understanding System," in this volume.