USCIPI Report 1010

IMAGE UNDERSTANDING RESEARCH

Semiannual Technical Report

Covering Research Activity During the Period

1 October 1980 through 31 March 1981

R. Nevatia
A.A. Sawchuk
Principal Investigators
(213) 743-5506

Image Processing Institute
University of Southern California
University Park
Los Angeles, California 90007
31 March 1981

This research was supported by the Defense Advanced Research Projects Agency and was monitored by the Air Force Wright Aeronautical Laboratories under Contract F-33615-80-C-1080, ARPA Order No. 3119

Wright-Patterson Air Force Base, Dayton, Ohio.

The purpose of this research program is to develop techniques and systems for understanding images, particularly for ... mapping applications. The research activity includes low level image analysis and feature extraction, statistical and structural texture analysis, symbolic image representations, and image to map correspondence using relational structures. Additional activity is concerned with VLSI architectures for implementation of image analysis and image understanding operations.

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

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER USCIPI Report 1010	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE *(and Subtitle)* IMAGE UNDERSTANDING RESEARCH		5. TYPE OF REPORT & PERIOD COVERED Semiannual Tech. Report 1 Oct. 80 - 31 March 81
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) (Principal Investigators) Ramakant Nevatia Alexander A. Sawchuk		8. CONTRACT OR GRANT NUMBER(s) F-33615-80-C-1080
9. PERFORMING ORGANIZATION NAME AND ADDRESS Image Processing Institute University of Southern California University Park, Los Angeles, Ca. 90007		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS DARPA Order No. 3119
11. CONTROLLING OFFICE NAME AND ADDRESS Defense Advanced Research Projects Agency 1400 Wilson Boulevard Arlington, Virginia 22209		12. REPORT DATE March 31, 1981 13. NUMBER OF PAGES 105 pages
14. MONITORING AGENCY NAME & ADDRESS*(if different from Controlling Office)* Air Force Wright Aeronautical Laboratories Wright-Patterson Air Force Base Dayton, Ohio 45433		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)*

Image Understanding, Scene Analysis, Image Segmentation, Edge Detection, Image Matching, Texture Analysis, Texture Synthesis, VLSI Systems, VLSI Processors, Residue Processors.

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

This technical report summarizes the image understanding and VLSI system research activities performed by the USC Image Processing Institute and the Hughes Research Laboratories during the period 1 October 1980 through 31 March 1981 under contract number F33615-80-C-1080 with the Defense Advanced Research Projects Agency, Information Processing Techniques Office. This contract is monitored by the Air Force Wright Aeronautical Laboratories,

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

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

ABSTRACT

This technical report summarizes the image understanding and VLSI system research activities performed by the USC Image Processing Institute and the Hughes Research Laboratories during the period of 1 October 1980 through 31 March 1981 under contract number F33615-80-C-1080 with the Defense Advanced Research Projects Agency, Information Processing Techniques Office. This contract is monitored by the Air Force Wright Aeronautical Laboratories, Wright-Patterson Air Force Base, Dayton, Ohio.

The purpose of this research program is to develop techniques and systems for understanding images, particularly for mapping applications. The research activity includes low level image analysis and feature extraction, statistical and structural texture analysis, symbolic image representations, and image to map correspondence using relational structures. Additional activity is concerned with VLSI architectures for implementation of image analysis and image understanding operations.

ACKNOWLEDGMENT

The manuscript for this report was produced on a Graphic Systems, Inc. C/A/T4 film output device using SCRIBE. We are thankful to Mr. Fabrice Clara for his assistance with SCRIBE formatting and to Ms. Hilda Marti for editing and production of the report.

TABLE OF CONTENTS

1. IMAGE UNDERSTANDING PROJECTS

1.1. Research Overview

This report primarily describes the results of our research under contract F-33615-80-C-1080, but also contains results of some research only partially supported by this contract. Our research includes work at many levels of an Image Understanding system, including low level feature extraction, symbolic descriptions and image matching. We have also been working with Hughes Research Laboratories on hardware implementation of IU algorithms using VLSI technology. The research results are summarized below.

Scene Matching

We have continued to work on the problems of matching two images of a scene, or one image with a map, by generating symbolic descriptions from each. We have implemented matching techniques. We feel that our system can handle complex aerial images, the major limitations being due to failures of the segmentation procedures. We have initiated work to use the map as a guide to such segmentation.

Symbolic Texture Analysis

We have a system in highly developed form for describing natural textures in terms of their primitives and relations among them. The descriptions consist of the primitive sizes and repetition patterns if any. A statistical analysis of our technique agrees with the experimental results. The descriptions have been tested for texture identification and we are investigating their use to estimate surface orientation form texture gradients.

Texture Synthesis and Analysis

We have been working on several different statistical techniques for synthesizing natural textures. Both gray level and binary textures can be synthesized, and ten distinct techniques with various tradeoffs have been explored. The tradeoff parameters include such factors as computation time for generation, computation time for data collection, memory requirements, and quality of simulation. Many commonly occurring natural textures have been adequately simulated using very simple models, providing potentially great information compression for many applications. Other textures with macrostructure and nonstationary characteristics require more extensive computation to synthesize realistic, visually pleasing results. Although the success of any synthesis method is highly dependent on the texture itself and the modeling scheme chosen, general guidelines for predicting the performances of various techniques have been developed.

We also hope to use these techniques for texture classification and image segmentation. Some preliminary experiments employing statistical feature selection and classification techniques for discrimination have been undertaken by this approach.

Other Projects

We are continuing to make improvements to our road finding and linear feature extraction programs. We have incorporated Laplacian-Gaussian masks, suggested by Marr, as an alternative for low level edge extraction. We have also implemented techniques of connecting segments to give more continuous boundaries.

We are also investigating the use of hierarchical gradient 1 techniques for matching of 2-D and 3-D shapes. We hope to present results in a later paper.

Hardware Implementation

In continuing work with Hughes Research Laboratories, Malibu, California, we are exploring architecture and hardware issues in the implementation of image understanding algorithms by VLSI techniques. The initial part of the study has concentrated on three algorithms: a) Nevatia-Babu Line Finder; b) Ohlander-Price Region Segmentor; and c) Laws 1 Analysis System (see section 2 for details) These three algorithms are all very computation intensive and have a broad range of applications in image understanding research. Common to algorithms a) and c) are extensive two-dimensional convolutional processing, especially in the early stages of the algorithm. This convolutional processing is largely local and is well matched to the nature of VLSI systems, in which interconnections are difficult to implement.

More recently, the convolution problem has led to a detailed design for a 5x5 pixel convolution processor based on residue arithmetic. The system is called RADIUS (Residue Arithmetic Digital Image Understanding System). The residue arithmetic approach has the advantages of modularity, ease of design, programmability, broad application to many problems, and ease of implementation in many integrated circuit technologies with submicron structures. Using residue arithmetic, there are no carries in the numerical computation and minimal interconnections on the chip are required. Very high speed processing is possible because many of the numerical operations reduce to table lookups in binary digital RAM's. Special purpose integrated circuits to perform part of the processing are being fabricated, and the construction of an experimental system is in progress. In addition to the increased computational speed for convolution, the processor has applications in evaluation of polynomial functions, integer coefficient transforms, enhancement operations, and moment calculations.

1.2. Relaxation Matching Applied To Aerial Images

K.E. Price

Introduction

We have developed a symbolic matching system which can be used for a variety of matching tasks in scene analysis. The system is designed to handle many of the problems encountered in the analysis of real scenes: noisy feature values, missing elements, extra pieces of objects, many features, many objects. At the heart of this system is a relaxation based matching scheme. A variety of relaxation procedures have been used with varying results.

The basic matching procedure [1] and the various relaxation methods [2,3] are discussed elsewhere and will only be outlined here. This paper will concentrate on a discussion of the overall matching system and the performance of the various relaxation techniques.

The input to the matching procedure is two relational structures-one for the model of the scene and the other for the input image. The structures are represented as graph structures with objects at the nodes (with associated feature values) and relations between objects as the arcs between the nodes.

Symbolic Matching Procedure

The goal of the matching procedure is to find the objects in the image (regions and lines) which best match the objects given in the model. This is essentially finding a subgraph in the image which is isomorphic to the model, except that objects may be missing and single nodes in the model may correspond to several in the image when objects are broken apart by the segmentation procedures.

The matching procedure is divided into two iterations (see Fig. 1). The outer loop consists of computing initial estimates of the assignments for all elements in the model. Up to 30 potential assignments for each element are maintained. Then a relaxation procedure is applied which updates the ratings of the assignments. When the rating for one assignment of an element in the model exceeds a threshold, the relaxation update loop is terminated and all assignments above the threshold are made permanent. Then the process continues with the recomputation of the initial estimates, but now relations (above, near, adjacent) with the permanently assigned elements can be used in the computation.

The repeated initialization is a crucial component of the process. Since the initial guesses are made only on the basis of feature value (color, shape, texture, etc.) many incorrect assignment are initially highly rated. The relaxation steps eliminate many of these mistakes, but cannot correct all of them

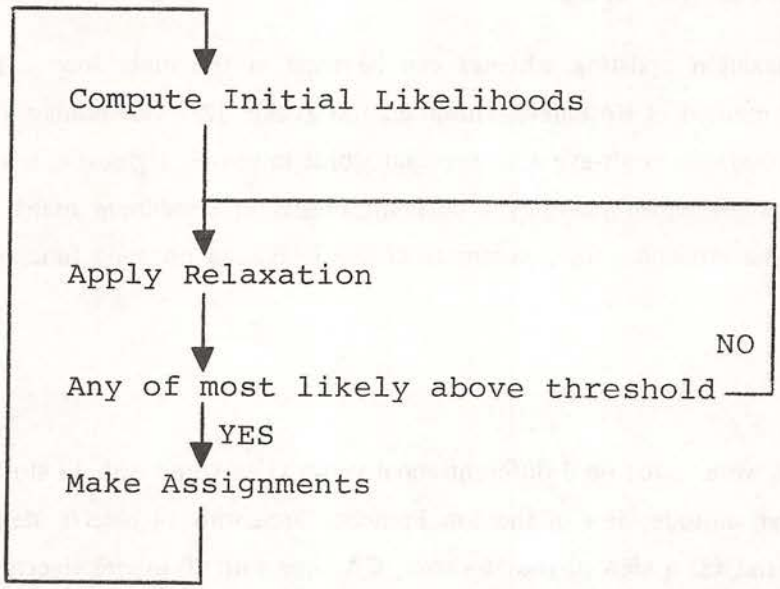


Fig. 1. Use of the relaxation procedure in the overall matching system.

because some correct assignments are not among the early candidates. This procedure also easily allows for multiple segments in the image to be assigned to one element in the model.

The final termination condition is the number of iterations without any assignments reaching the threshold. This number must be large enough so that valid assignments can reach the threshold but not so large that an incorrect assignment is forced, by default, to a large value. This is especially true with our primary relaxation method [1] where something is always forced to a high value (how rapid can be controlled and we use a relatively fast setting).

Several different relaxation updating schemes can be used in the inner loop. The simplest technique is the "classical" method of Rosenfeld, Hummel, and Zucker [2]. Our primary technique [1] is similar, except that the updating is always in a direction which improves a global criterion. A third method is that of Kitchen [3] which provides a different means of combining match rating from different properties. See the Appendix for a summary of the relaxation updating functions for these methods.

Results

The various methods were tested on 3 different aerial views (1) a scene with 14 storage tanks, of 5 different sizes, (2) a high altitude view of the San Francisco area with 14 objects identified in the model (95 in the image), and (3) a view of the Stockton, CA. area with 20 model objects (24 or more valid matches possible, and more than 200 image elements). The results on these images allow us to make some general comments on the performance of the various methods.

The basic technique of Rosenfeld et al. [2] quickly makes several assignments in each of the test scene, but then reaches a stable state with low probabilities for the most likely assignment.

The method of Kitchen [3] allows for a variety of combining methods (by using different functions for the fuzzy set operations). Additionally, restricting the computation to use only the one most likely assignment of neighboring (related elements reduces the time substantially and improves the performance.

The results of running the various relaxation methods are presented in Table 1. The set of assignments shown n Figs. 2, 3, and 4 are the ones generated by the criteria optimization method described fully in [1] (FP in Table 1). K-1 is the Kitchen method [3] which uses MIN, MAX, and the mean for $\cap$, $\cup$, and the outer $\cap$ (Form 4 in his paper), the use of MIN for the outer (Form 1 in his paper) produced no assignments in our tasks. K-2 uses product instead of MIN (Form 5). K-3 is the same as K-1 exact all possible assignments of neighbors are considered rather than the one most likely assignment. Kitchen's Form 6 produced the same results on our tasks as K-2. RHZ is the classical

Table 1. RELAXATION RESULTS.

Method	Scene	Number Correct	Incorrect	Not Assigned	Time
K-1	1 (Tank Farm)	12	0	2	5
K-2	1	14	17	0	30
K-3	1	0	0	14	-
RHZ	1	14	28	0	29
FP	1	14	0	0	4
K-1	2 (San Francisco)	7	0	7	19
K-2	2	9	0	5	28
K-3	2	7	0	7	4:50
RHZ	2	8	11	6	53
FP	2	14	0	0	9
K-1	3 (Stockton)	9	0	11	1:25
K-2	3	16	4	5	3:00+
K-3	3	6	0	14	6:00+
RHZ	3	16	11	6	2:30+
FP	3	24	1	0	1:00

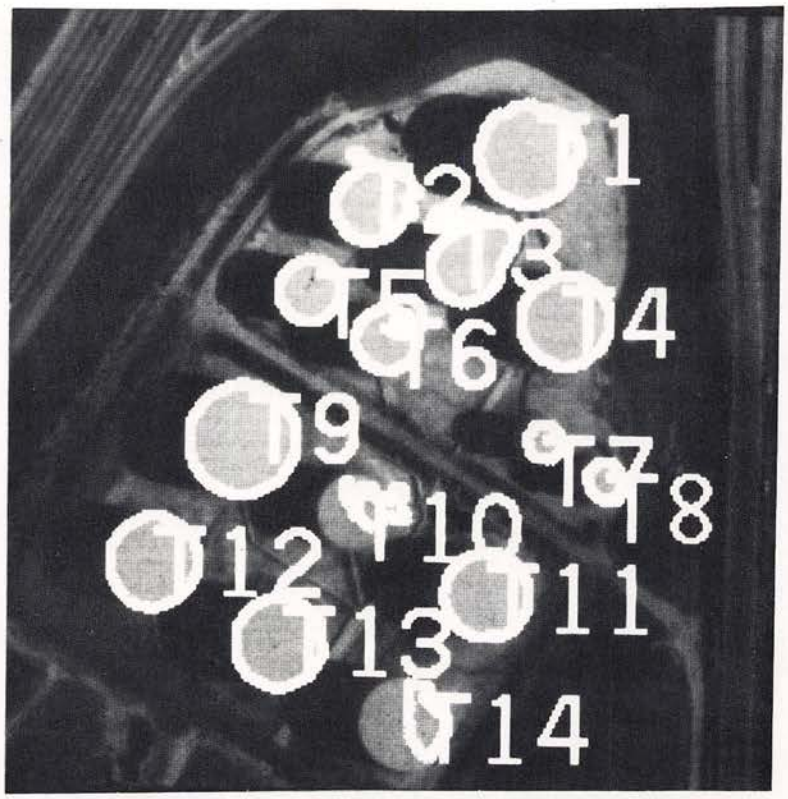


Fig. 2. Storage tank results.

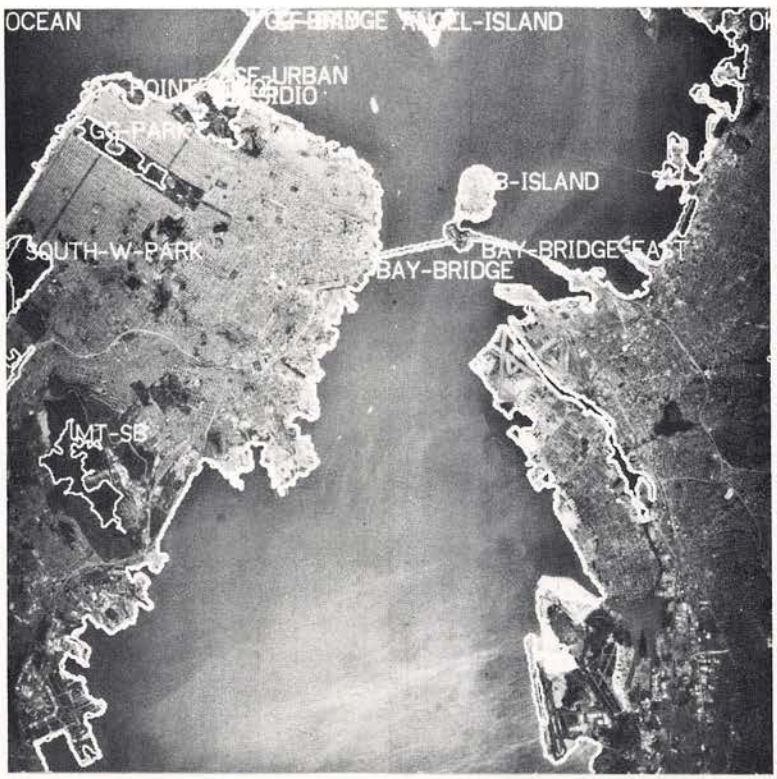


Fig. 3. San Francisco scene results.

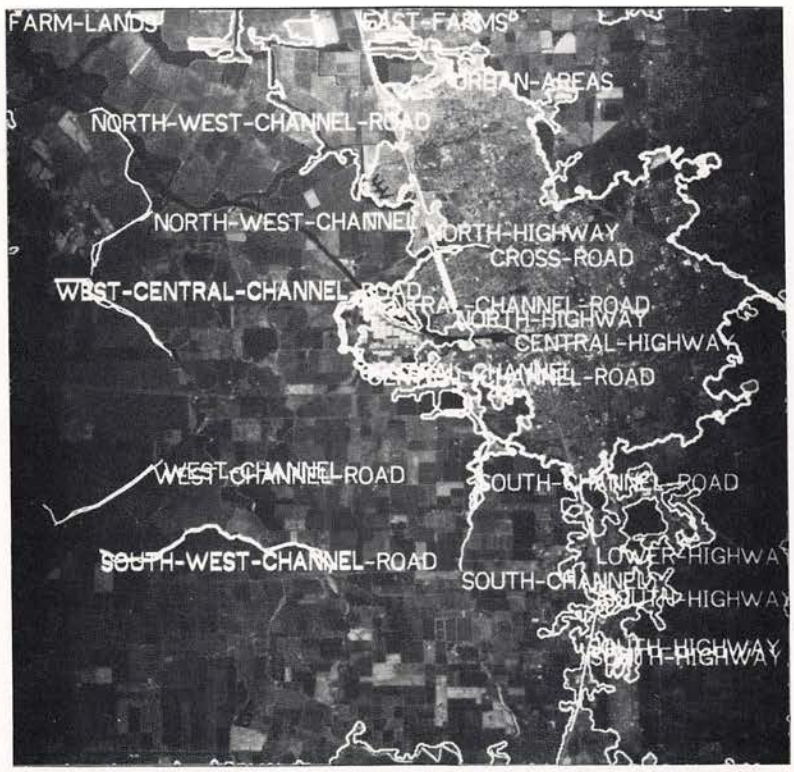


Fig. 4. Stockton area results.

Rosenfeld et al. method [2] and is included as a historical reference point.

The threshold for forcing a permanent assignment is 0.75 (for the Kitchen algorithms the values were normalized only for this test) and 15 iterations were run before terminating the procedure due to lack of assignments. In tests with our algorithm, changing the 0.75 threshold (between about 0.7 and 0.8) by small amounts has little or no effect. Increasing it requires more iterations and thus more time and some assignments may be lost. The results include only those assignments which exceeded the threshold (0.75) and does not include the most likely assignments at the time of termination (15 iterations). Including all these would increase the number of correct ones with no clear separation in likelihood values between correct and incorrect ones.

Several comments can be made from these results. First, Kitchen's method was not designed to fit into our matching system and is not oriented toward quickly producing unambiguous results for a few of the elements in the graph. But this feature is necessary when dealing with problem domains such as these (i.e. many feature values, many elements, similar objects, and noisy data).

Second, considering more alternatives for neighbors does not improve performance, but decreases it with a substantial increase in time. this was not totally obvious without experiments) since the likelihoods of the second, third and other alternations are much less than the most likely one they should contribute little, it any, to the computation. Tests with the other methods (K-2, FP) give similar results (decreased performance increased time).

Third, our method performs better than the others in these tasks. One major reason is the optimization procedure used to guide the updating thus some assignments are discovered early and then contribute substantially in the search for further matches. Also, in our system, all relations and features contribute to the rating rather than only the best or worst as with Kitchen using MIN and MAX.

Fourth, the 17 incorrect assignment by K-2 for Scene 1 start after 13 correct matches are located. This is accounted for by the way we handle multiple assignments for model elements and by the fact that multiple matches for image elements are only discouraged not forbidden. The change from MIN to product (between K-1 and K-2) also contributes to the problem.

The updating approach adopted by Faugeras [1] clearly performs better and operates faster. Each iteration of this program takes longer, but fewer are required.

References

[1] O.D. Faugeras, K. Price, "Semantic Description of Aerial Images Using Stochastic Labeling," *IEEE-Trans PAMI*, to appear, preliminary version in Proc. Image Understanding Workshop, Univ. of Maryland, April 1980.

[2] A. Rosenfeld, R.A. Hummel, and S.W. Zucker, "Scene Labeling by Relaxation Operations," *IEEE Trans-SMC*, Vol. 6, No. 6, pp. 420-453, June 1976.

[3] L. Kitchen, "Relaxation Applied to Matching Quantitative Relational Structures," *IEEE Trans.-SMC* Vol. 10, No. 2, pp. 76-101, Feb. 1980.

Appendix

We only present a summary of the equations for the relaxation updating the details are contained in the appropriate papers. Therefore many of the terms will not be fully explained. The classical Rosenfeld et al. method [2] is:

$$p_i^{(n+1)}(n_k) = \frac{P_i^{(n)}(n_k)Q_i^{(n)}(n_k)}{\sum_{n_l \text{ in } N} p_i^{(n)}(n_l)Q_i^{(n)}(n_l)} \tag{1}$$

$p_i(n_k)$ is the likelihood of assignment for unit i to name k, Q is a measure of the compatibility of the assignment with assignments of neighboring units. N is the set of all possible names (image elements).

The Kitchen updating method [3] is (rewritten for our problem)

$$p_i^{(n+1)}(n_k) = LC_i(n_k) \tag{2}$$

$$LC_i(n_k) = \left[\bigcap_{\substack{\text{all relations rel} \\ \text{s.t. } u_i \text{ rel } u_j}} \left[\bigcup_{\substack{\text{all } n_l \\ \text{s.t.} \\ n_k \text{ rel } n_l}} \left[p_i(n_k) \cap p_j(n_l) \cap c(u_i,n_k,u_j,n_l) \right] \right] \right] \tag{3}$$

$$\cap = \left[\bigcap_{\substack{\text{all features} \\ \text{of } u_i}} \left[p_i(n_k) \cap f(u_i,n_k) \right] \right]$$

where $\cap$ and $\cup$ are the fuzzy logic AND and OR operations. C is a local consistency measure which is also used to compute the Q's in the equations above and below. The generality of the original formulation is not retained - only features of single objects and relations between two objects are given - not the arbitrary number of the original.

The third method of Faugeras [1] is:

$$\vec{p}_i^{(n+1)} = \vec{p}_i^{(n)} + \rho_n P_i\{\vec{g}_i^{(n)}\} \qquad (4)$$

where ρ_n is a positive number (step size) to control the speed, P_i is a projection operator to maintain the constraint that $\vec{p}_i$ is a probability vector, and $\vec{g}_i$ is a gradient function computed from the compatibility measures and current probability values for an object and its neighbors.

1.3. An Algorithm to Group the Edges of a Picture Using a Local Criterion

Gerard G. Medioni

The first step in the process of scene analysis is usually to extract the edges from the picture, along with some information about these edges, such as orientation, strength and relative brightness. Nevatia, Babu and Huertas have been using such techniques in the past at USC [1]. A good continuation would be to try to organize these edge elements into classes before going to the semantic processing of "making sense" out of them, that is labeling them with a list of reference tags. Of course, this classification is very subjective and the probability of misclassification is non-zero, but it requires no *a priori* assignment of probabilities and is hence information preserving; that is, even after classification, no final assignment or interpretation are done. In the case of aerial images, we decided to take advantage of the following properties of roads and other man-made linear constructions:

- □ In most cases, a road is locally defined by two antiparallel segments
- □ Also, a road is locally straight.

Following these two simple and reasonable rules, we developed an algorithm taking a pair of antiparallel line segments as input and producing a continuation of these segments in both directions, creating new segments where gaps are found. The core of the algorithm is

Given a segment, create a search window, scan it and extract the most likely continuing segment, if any; repeat until no segment can be further expanded.

The dimensions of the search window are determined by the length L and the angle ϕ. Let θ_1 be the angle of the current segment and θ_2 the angle of the prospective continuing one, we choose the closest segment to the axis A_1B_1 that minimizes $|\theta_2-\theta_1|$, provided that this value does not exceed a certain threshold (30^o). In the figure, the search window is thick-lined.

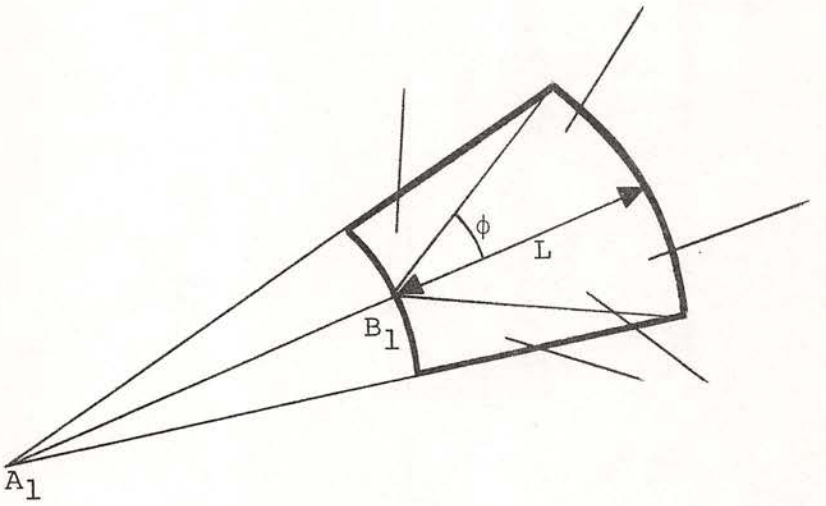


Figure 1.

The only parameters to this algorithm are L, the neighborhood length and ϕ, a representation of its width. It was found that the algorithm is not very sensitive to variations of these parameters, so that the values L=30 pixels and ϕ =10 degrees are very good default values. The running time of the algorithm is dependent upon the total number of segments present in the picture and upon the size of the window. It took 30 secs to process Fig. 2(b). In Fig. 2, (a) represents the original image, (b) the edges and (c) the continuation of some segments. As demonstrated on this example, the results are very encouraging. The future developments of this study are: In a first step, expand all pairs of segments that are the left and right components of an antiparallel system in an image and therefore obtain a set of "SUPER-SEGMENTS" as a basic working set, then express the model of the scene we are looking at in terms of these super-segments, and finally find a functional relationship between elements in both "graphs", that is find a semantic labeling of these super-segments.

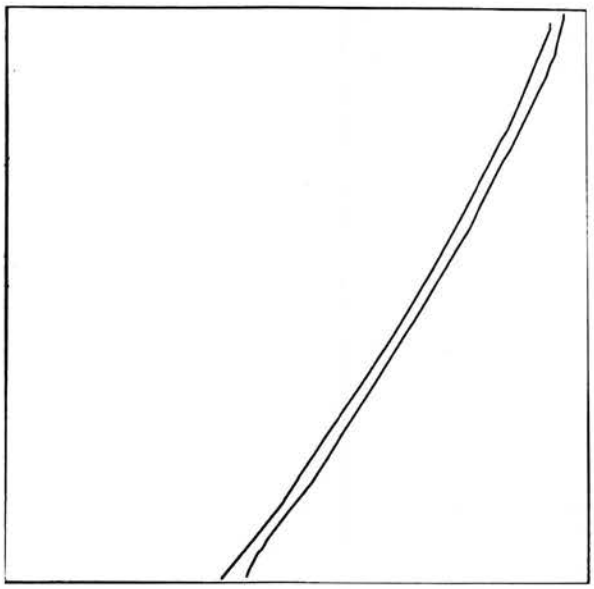
References

[1] R. Nevatia and K. Ramesh Babu, "Linear Feature Extraction and Description," *Computer Graphics and Image Processing*, Vol. 13, June 1980, pp 257-269.

[2] D. Marr, "Early Processing of Visual Information," *Phil. Trans. Roy. Soc.* B.275, 1976.

(a) Original image 256x256

(b) Extracted segments



(c) Result of the algorithm

Figure 2.

1.4. Edge Detection in Aerial Images Using $\nabla^2 G(x,y)$

A. Huertas **and** *R. Nevatia*

Introduction

The use of an orientation-independent operator for the purpose of edge detection in aerial images is discussed. The operator is $\nabla^2 G(x,y)$, the Laplacian of a Gaussian distribution.

Intensity changes are detected by finding the zero-crossings in $\nabla^2 G(x,y)*I(x,y)$ for image I. Different kinds of features are seen with varying accuracy by filters of varying sensitivity. The sensitivity of the filters is primarily associated with **w**, the width of the central excitatory region of the operator, and secondly, with λ the diameter of the operator. The smaller the value of **w**, the more sensitive the filter and the more detail is seen, including noise.

The accuracy of the filters is limited by the size of the individual features in the image and by the spatial separation between them in relation with the width of the central excitatory region of the filter.

The zero-crossings found for each filter are assigned a magnitude and a orientation for further linking along continuous lines of zero-crossings. Only one convolution operation is required for each filter and no thinning or thresholding is required. Additional information on the $\nabla^2 G$ filters can be found in [1].

Goal-oriented processing can use several available parameters and simple measures associated with each zero-crossing to decide the "goodness" of each zero-crossing for the intended purpose. It is shown that a straightforward global combination of filtered images oversees the excessive detail seen by the most sensitive filters while augmenting the view of the less sensitive ones with unseen detail. Results obtained for an aerial image are presented.

Combined zero-crossing images are the input to a higher level feature detector to be described in a subsequent report. It uses the combined output as a basis to perform goal-oriented processing with feedback access to individually filtered zero-crossing images.

Detection of Intensity Changes

Wherever an intensity change occurs, there is a peak in the first directional derivative of the intensity and a zero-crossing in the second directional derivative. Detection of intensity changes is then reduced to finding the zero-crossings in the second derivative of the intensity in the direction of maximum slope at the zero-crossing. By using an orientation-independent operator the number of required convolutions is reduced to one and the direction that yields the maximum slope is found locally. This direction is used to compute a magnitude and an orientation to be associated with each zero-crossing. Basically the magnitude is the value of the largest slope measured at the zero-crossing among a selected number of directions, and the orientation is the direction perpendicular to the direction of maximum slope. The convention followed for orientation is that the bright side of the edge lies to the right of the line of zero-crossings.

Figure 1 shows a profile of the operator and the parameters involved in the following discussion:

a) $G(x) = 1/(\sigma(2\pi)^{1/2}) \cdot \exp[-x^2/(2\sigma^2)]$

b) $G'(x) = -x/(\sigma^2(2\pi)^{1/2}) \cdot \exp[-x^2/(2\sigma^2)]$

c) $G''(x) = -1/(\sigma^3(2\pi)^{1/2}) \cdot (1-(x^2/\sigma^2) \cdot \exp[-x^2/(2\sigma^2)]$

In two dimensions:

d) $\nabla^2 G(x,y) = -1/(2\pi\sigma^4) \cdot (2-((x^2+y^2)/\sigma^2)) \cdot \exp[-(x^2+y^2)/(2\sigma^2)]$

$w = 2\sigma$ (Width of the excitatory region of the filter).
$\lambda = 2\pi\sigma$ (Diameter of the filter).

Figure 2a shows the intensity profile of an ideal step edge taken in the direction dmax for which the slope measured at the corresponding zero-crossing shown in Figure 2b is maximum. Several parameters associated with each zero-crossing are also shown. They can be used to determine the "goodness" of a zero-crossing for a given purpose.

The following criterion has been adopted to determine the existence of a prospective edge point at a zero-crossing. Basically we require that the amount of energy on both sides of the zero-crossing be nonzero. That is a1 and a2 in Figure 2b are nonzero in absolute value. This criterion,of course, does not prevent false or displaced edge points to be rejected but eliminates ghost echoes caused by specific physical phenomena as well as very small and faint edges.

To See Or Not To See With $\nabla^2 G(x,y)$

The sensitivity of the filter depends primarily on the value of **w**, the width of the excitatory region of the filter. To a lesser extent it also depends on λ, the diameter of the filter. The diameter of the filter is about $3w$ (πw, precisely). The smaller the value of **w** the smaller the value of λ, the more detail is seen, and the sensitivity to noise is increased. Filters with larger values of **w** see less detail and

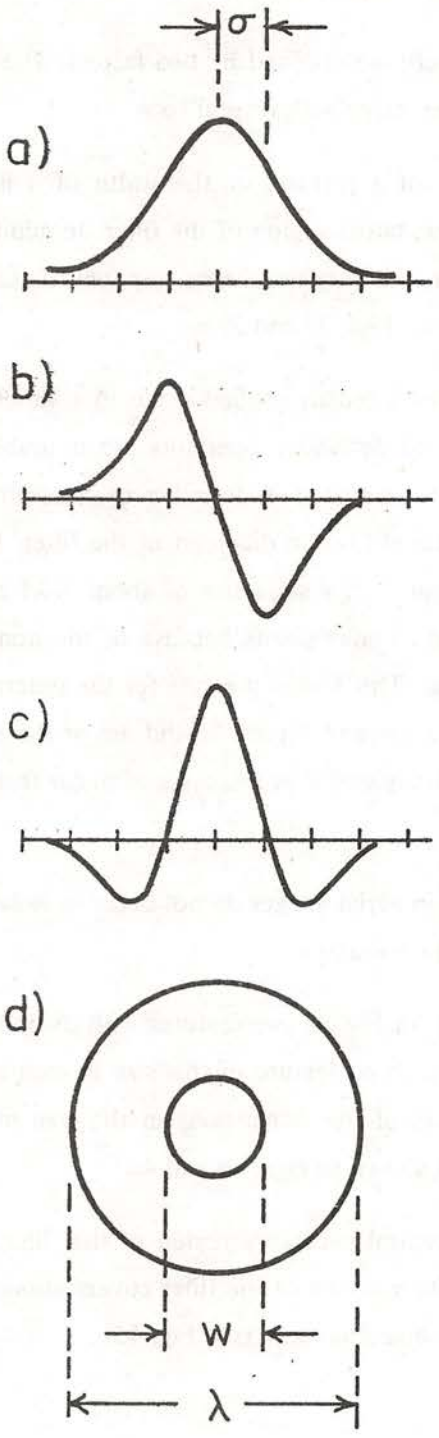


Figure 1.

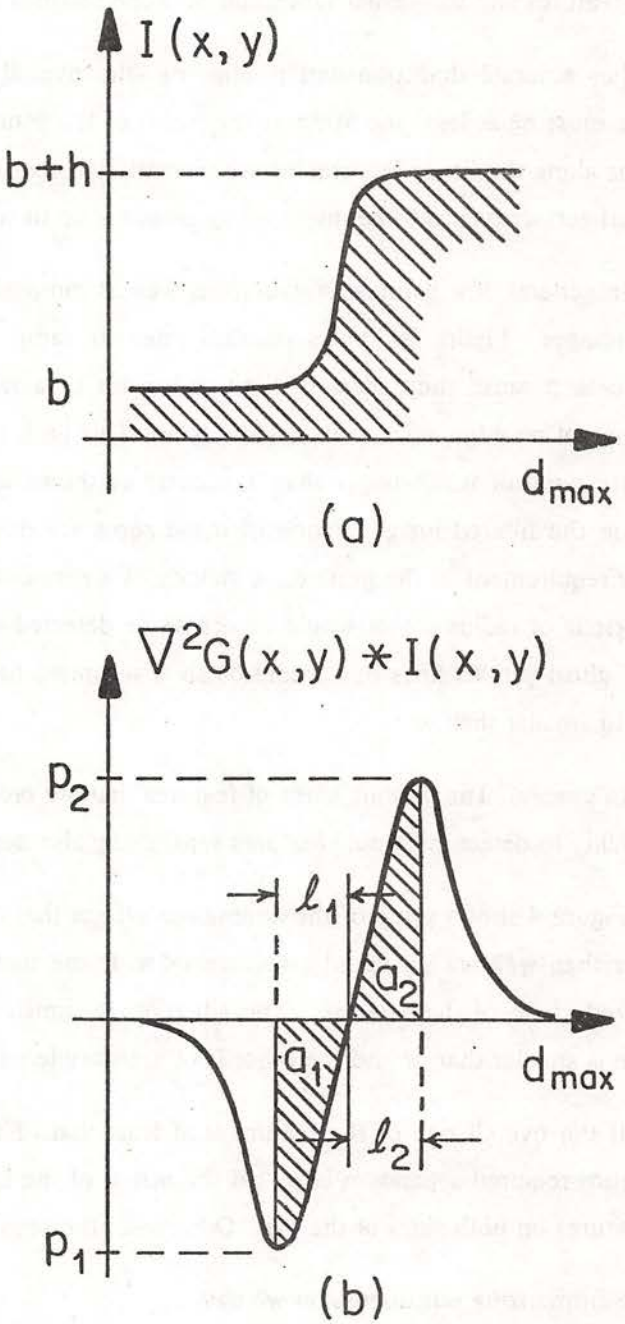


Figure 2.

are much less sensitive to noise due to the smoothing properties of the Gaussian.

The ability to detect accurately located edge points is chiefly determined by two factors: The size of the features and the spatial separation between features in the image with respect to **w**.

For accurate detection and positioning, the overall size of a feature, or the width of a linear feature must be at least the same as the width of the central excitatory region of the filter. In addition, sections along the linear features where intensity changes occur in a very non-linear way, will cause the detected zero-crossings to be displaced by amounts up to **w/2** (see Figs. 3a and 3b).

In general, low contrast situations as well as ramp and roof intensity profiles occur in a variety of aerial images. Figure 3d shows an ideal intensity ramp. Second derivative operators are desirable in these case because they allow to limit the width of a ramp for which it is desirable to consider the existence of an edge point at its middle point. This limit is precisely λ, the diameter of the filter. If an intensity ramp of width larger than λ occurs, as shown in Figure 3d, a sequence of about $(l-\lambda)$ zeros result in the filtered image. None of these zeros are detected as edge points because of the nonzero energy requirement in the immediate vicinity of a zero-crossing. This is also the case for the outermost ghost circle of radius λ that would otherwise be detected in the cases of Figure 3a and 3c; or the extra pair of ghost parallel lines that would be seen separated by a distance of λ in the cases of linear features of width smaller than **w**.

In general, The various kinds of features that are present in aerial images do not occur in isolation. The ability to detect individual features separatedly also depend on **w** and λ.

Figure 4 shows some of the anomalous effects that occur. In Fig. 4a two features with overall size smaller than **w/2** are separated a distance $s<w$. Recall that an isolated feature of this size is seen as an oversized circle of diameter **w**. The situation is similar if one of the dimensions in the size of the feature is smaller than **w** and the other is of arbitrary length, as shown in Figs. 4b and 4c.

If the overall size of the feature is at least that of the central excitatory region of the filter, the minimum required separation is **w/2** if the action of the inhibitory region of the filter covers enough of the features on both sides of the gap. Otherwise, the required separation is **w** (see Fig. 4d).

Summarizing our discussion we have:

1. On size:
    a. If the overall size (diameter) of a feature is smaller than **w**, the diameter of the excitatory region of the filter, the detected edge points form either a ghost or a oversized circle shaped blob. The amount of oversize is (**w/2** - r), r being the radius of the feature.

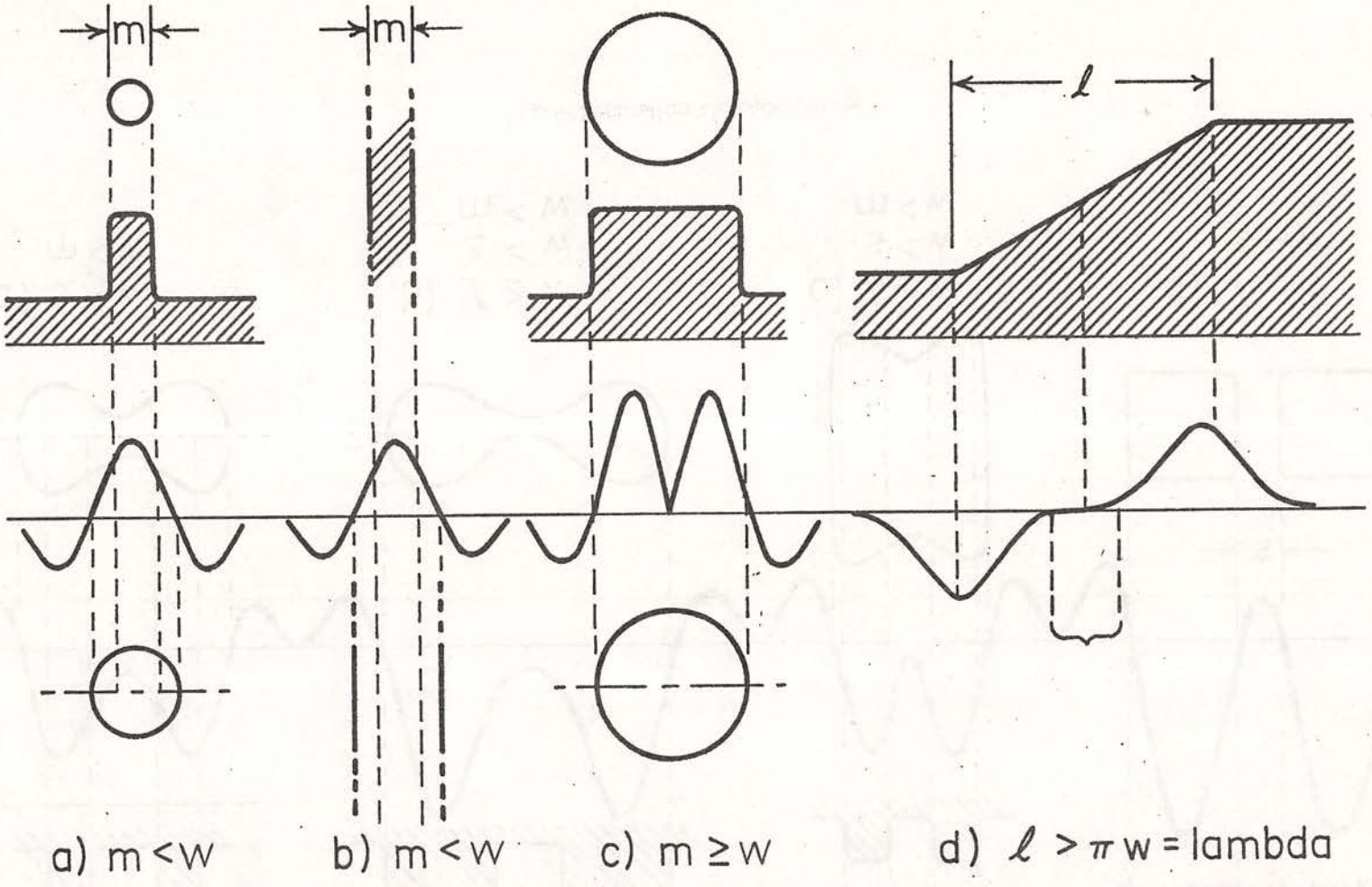


Fig. 3. Size considerations.

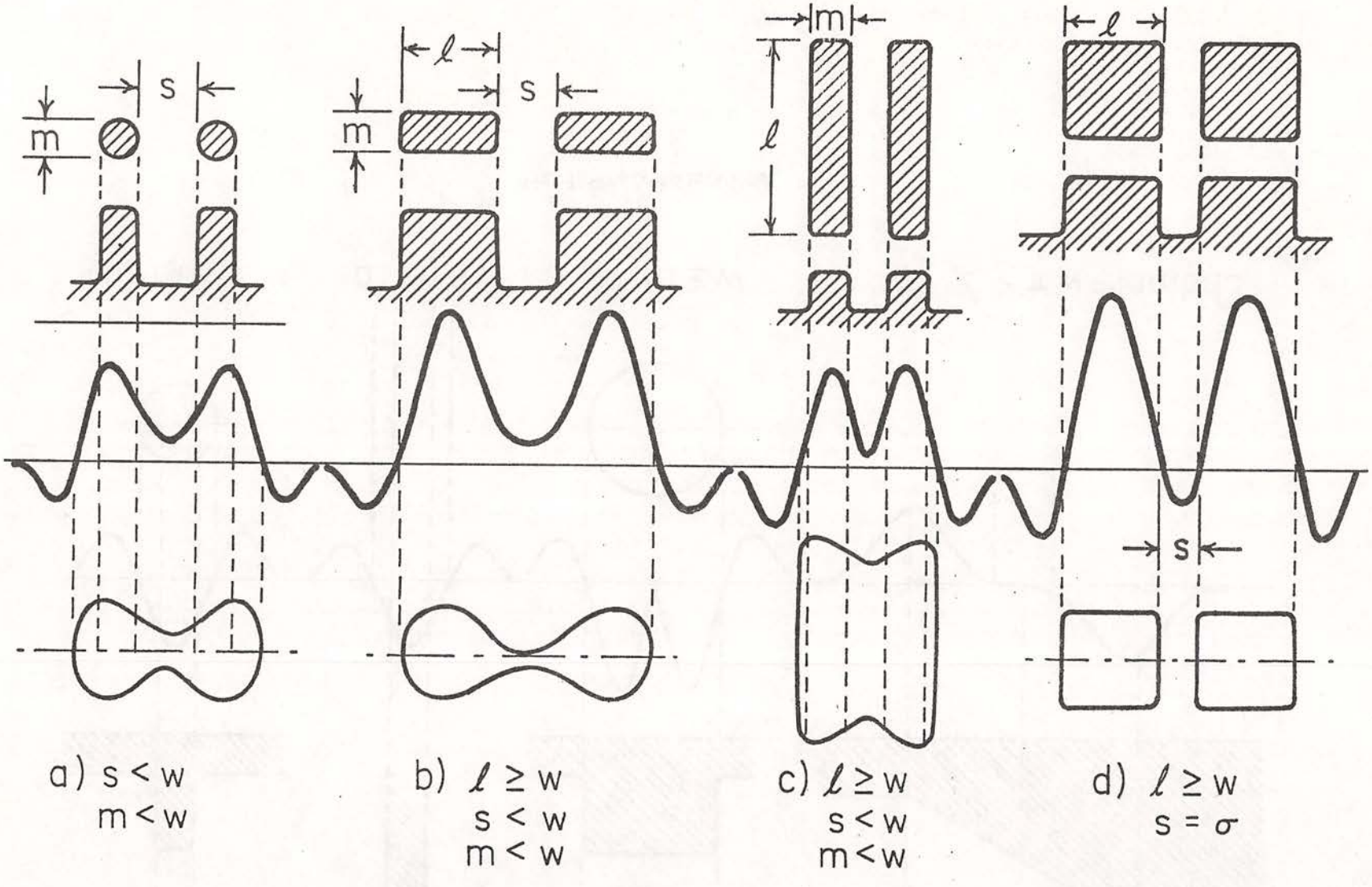


Fig. 4. Separation Considerations.

b. If one of the dimensions (length) of the feature is larger than w and the other (width) is smaller than w, the detected edge points form an elongated blob. The length of the blob is the same as the length of the feature, but the width is increased by 2*(w/2 - l), l being the width of the feature.

c. If the overall size of the feature or its smaller dimension is at least w, the detected edge points depict the feature very closely in shape, size and location. Exceptions are sharp corners and end points which tend to be rounded up.

2. On separation:

a. If the separation between two features of diameter l<w is smaller than w, the correspondingly detected ghost or oversized closed contour of zero-crossings appear blended together into a larger ghost or oversized contour of zero-crossings of size w*(w+l). The minimum required separation for separate detection is then w.

b. If one of the dimensions l (length), of a feature is larger than w, and the other (width) is smaller than w, and the features are collinear or nearly collinear, the minimum distance for separate detection is w as in the previous case. The size of the blended contours of zero-crossings if the separation s is less than w , is w*(2*l+s). If the features are parallel or nearly parallel, the minimum required separation is also w. Otherwise the contour of zero-crossings formed is a blob of length equal to the length of the feature and (2*l+w/2+s) wide.

c. If the overall size (diameter) or the smallest dimension of the feature is at least w, the two separatedly detected contours of zero-crossings are very close in size and shape to those of the features in the image. As mentioned before, sharp corners and end points are rounded.

Experiment and Comments

Aerial images in general contain regions of natural texture such as forests and artificial or man-made textures such as low resolution urban areas. Both the size and the separation between individual texture features impose limits to the accuracy with which these features are seen, even if small values of w are used in the filters.

Texture features in general are seen as small closed curves in the zero-crossing image. Less sensitive filters make this anomaly even more severe by blending together groups of features which are seen as a single larger closed curve. Linear features such as roads consist of regions of approximately uniform intensity. Such property makes them consistently visible in the zero-crossing images obtained for filters of varying sensitivity. Other man made objects such as storage tanks and buildings that are located against contrasting backgrounds are likewise consistently detected with little or no change in location, size or shape. In less contrasting situations or at lower resolutions, the objects tend to blend with the background and observable changes in size and shape occur but their presence is consistent.

A desirable property of the detected lines of zero-crossings is their continuity along the curves in the zero-crossing image. This property greatly facilitates the linking of adjacent zero-crossings along these lines and consequently, the latter fitting of linked zero-crossings into linear segments, using

techniques like the one described in [2]. For less sensitive filters on the other hand, because of the smoothing properties of the Gaussian, sections of low contrast along a road, for example, will tend to blend together with nearby features in the zero-crossing image. Continuity therefore will not cause a line of zero-crossing to stop when the edge of the road is not discernible, but rather it will continue around a nearby feature before comming back on track. Goal-oriented edge detection can solve this problem by using evidence gathered by more sensitive filters as we will see in the next section.

Figure 5 shows the picture of a road and the zero-crossing images (linked) obtained with three filters of varying sensitivity: a) original; b) $w=3$; c) $w=4$ and d) $w=6$. Notice the detail seen by the most sensitive filter as well as the continuity and consistency with which the main road is seen. The size of the image is 384x256 pixels. The convolution with a 17x17 filter takes about 68 seconds on a DEC-10 under TENEX using an array processor. The detection of zero-crossing takes about 7 seconds on a DEC-10 under TOPS-20.

Notice that for the most part, it is not easy to tell the small buildings from the surrounding texture. Higher resolution is required to see buildings and other similar objects. Notice also the difficulty in dealing with highly textured images. Only the most sensitive filter may provide some information about the concentration and size of the texture features.

In general, fine detail imposes a problem. The more sensitive filters see it more accurately but the less sensitive ones provide better orientation information along long lines of zero-crossings. Goal-oriented processing can manipulate the trade-offs involved in a local as well as in a global manner. That is, the zero-crossing images can be individually filtered on several available parameters, or combined in a suitable manner.

Combination of Zero-Crossing Images

In this section we discuss one of various ways of combining zero-crossing images. The general idea is to oversee the fine detail seen by the most sensitive filters while augmenting the view of the less sensitive ones with unseen detail. Marr [1] suggests a spatial coincidence assumption that is basically a local one. It states that provided the zero-crossing from independent channels of adjacent size coincide, they can be taken together. That is, provided that the channels (at least two) are reasonably separated in the frequency domain, and their zero-crossing agree, the combined zero-crossing can be taken together to indicate the presence of an edge in the image.

The combination of zero-crossing images based on a coincidence criterion is difficult to achieve for aerial images. The increased amount of fine detail and subtleties prevent the proper registration of corresponding zero-crossings. The lack of registration is overcome by combining the filtered images

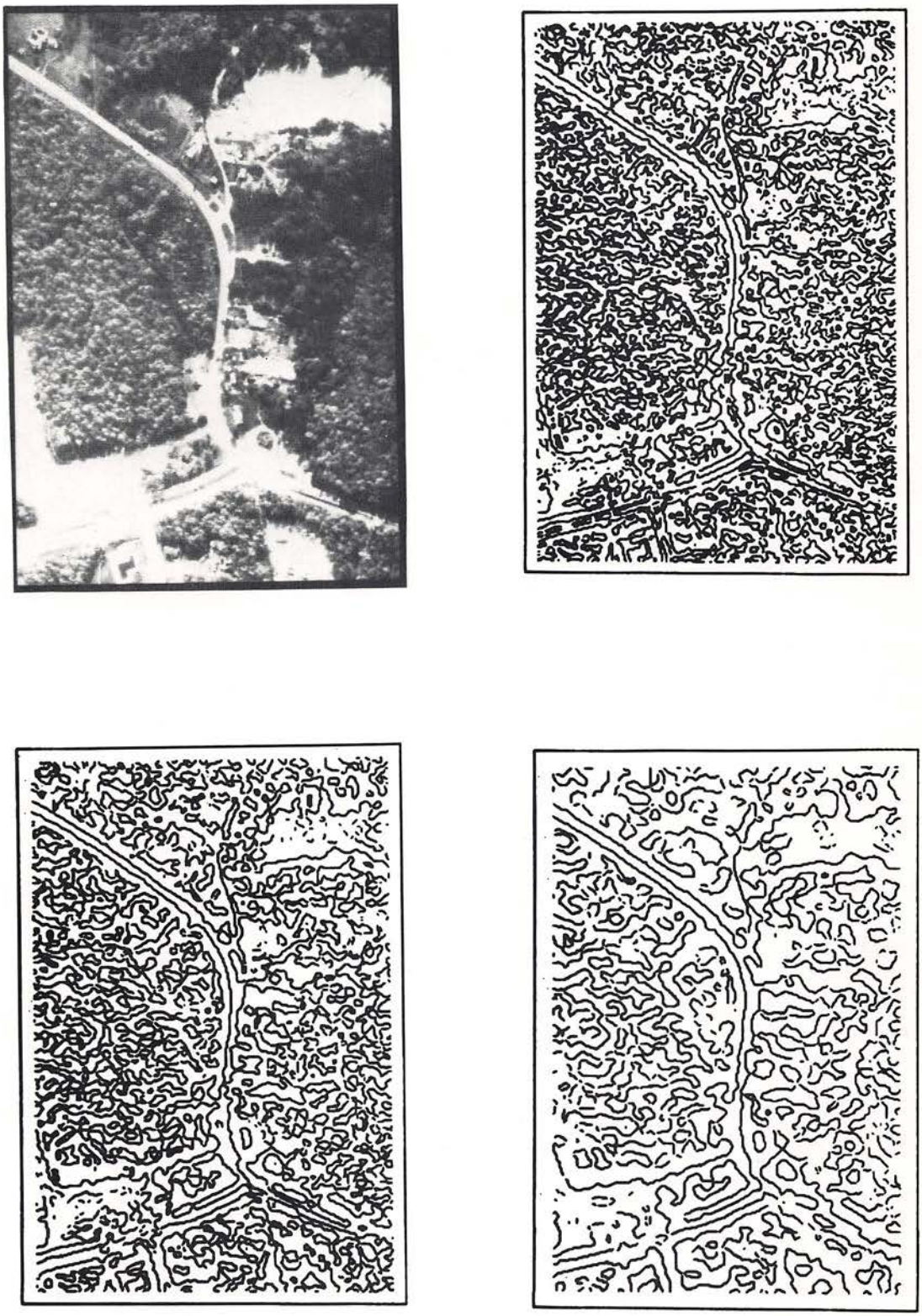


Fig. 5. Road: Zero-Crossing Images.

rather than the zero-crossing images. Coincident or nearly coincident zero-crossings will see their energy on both sides of the zero-crossing enhanced. Roughly coincident ones tend to average their energy and less coincident ones tend to cancel each other. In addition, the adjacency in the sizes of the filters is less tight using this global combination approach.

Figure 6a shows the result of the combination of the zero-crossing images individually obtained for $w=3,4$ and 6 (see Figure 5). Note that global boundaries are preserved while fine detail suppressed. In particular, The edges along the main road are less smooth for $w=3$ because the orientation of the individual zero-crossings changes at a faster rate. For $w=6$ the edges along the road are smooth but some of the narrow secondary roads are not seen. In the combined zero-crossing image the edges of the road are smooth and the secondary roads are present. Figure 6b shows the result of line segment fitting using the technique described in [2] and applied to the zero-crossing image of Figure 6a.

The value of the combined result depends on the intended purpose for the processing. A higher level feature detector and extractor that uses the combined output as its basis for the detection of cultural features is currently under investigation.

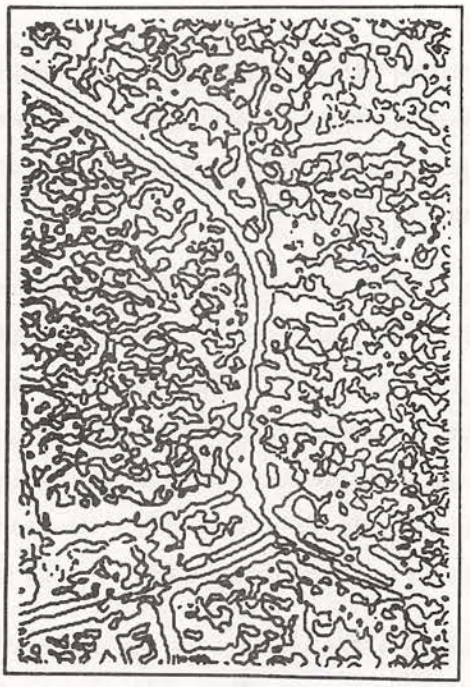
Conclusion

Fairly good results are obtained with the $\nabla^2 G(x,y)$ filters for a variety of aerial images. Although their limitations are explicit when fine detail and small features are involved, man made objects such as roads, channels, airstrips and buildings at a proper resolution are seen fairly well. The reduced number of convolution operations is advantageous specially when large images are being processed. No thinning or thresholding steps are required and small windows are sufficient for their detection. The continuity of the contours of zero-crossings facilitate the linking and the subsequent line fitting steps for higher level processing. Goal-oriented edge detection has available several parameters that can be used to measure the "goodness" of a zero-crossing for the intended purpose. Global combination of filtered images as discussed is straightforward and the combined output appears suitable for processing by higher level feature detectors and extractors.

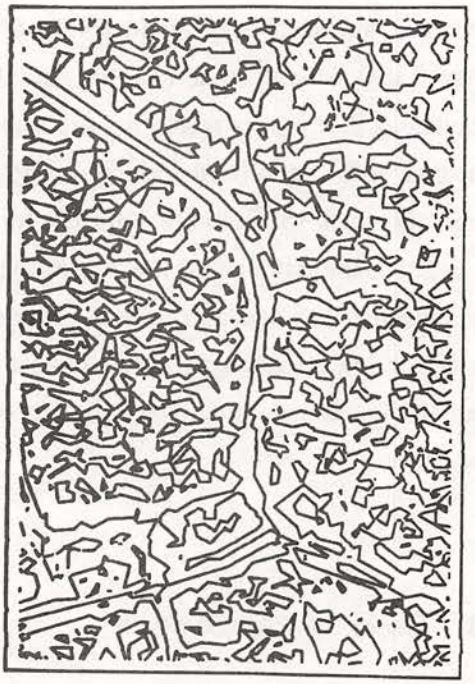
References

[1] D. Marr and E. Hildreth, "Theory of Edge Detection," *Proc. Royal Soc.* London, Vol. B, pp. 187-217, 1980.

[2] R. Nevatia and K.R. Babu, "Linear Feature Extraction and Description," *Comp. Graph. Imag. Proc.* , Vol. 13, pp. 257-269, 1980.



(a) Combined zero-crossing image with w = 3,4 and 6.



(b) Line segments.

Fig. 6. Road: Combined Zero-Crossing Image.

1.5. Automatic Grid Relation Extraction and Texture Reconstruction for Homogeneous Regular Textures

F. Vilnrotter,[1] *R. Nevatia,* **and** *K. Price*

Introduction

In a previous USCIPI Semiannual Report [1] a method for automatically determining the spatial relationships between texture primitives in homogeneous regular textures was described. Also, a possible approach for the automatic extraction of a minimum set of relationships (not necessarily unique) needed to characterize the underlying grid pattern in these types of textures was discussed. In the following the actual method used for automatic grid relation extraction is described and results are reported.

Determining the underlying structure of square or rectangular cellular textures has been investigated by Davis [2]; computing spatial relationships between connected regions of constant gray level has been investigated by Maleson *et al.* [3] and by Matsuyama *et al.* [4] who use "regularity vectors" to determine texture primitive arrangement.

Background

In previous reports, we have described programs used to generate descriptions of natural textures [5-6] and extract and describe texture primitives [7] and the spatial relations between them [1]. Short descriptions of some of these programs may also be found in [8-9].

In the first part of the program edge repetition arrays are produced using the edge and direction images corresponding to the original image (an edge repetition array is the binary case of a gray level co-occurrence matrix). These arrays are calculated for 6 directions (0, 30, 60, 90, 120, and 150 degrees), for both dark and light intensity objects, and distances within a range specified by the user. A comprehensive discussion of edge repetition arrays is given in [5].

In the second part of the program the edge repetition arrays are analyzed to determine whether there are predominant element sizes in any of the 6 scan directions and if so whether these elements occur at regular intervals within the image. The details of this analysis are presented in [6] and will not

---

[1]Felicia Vilnrotter is supported by a Hughes Aircraft Company Doctoral Fellowship.

be repeated here.

In the third part of the program groups of texture primitives are extracted and described, using the texture descriptions generated in part 2. Primitives are represented as masks. These masks are then individually analyzed to determine the characteristics (average size, intensity, etc.) of each primitive group. The details of the primitive extraction and description processes are given in [7].

In the fourth part of the program a set of placement rules is generated using the primitive masks generated in part 3. These placement rules are the most frequently occurring inter- primitive spatial relationships. A detailed discussion of the algorithm used to generate these rules is given in [1]. These spatial relationships, or placement rules are of the form:

PRIM1 = A PRIM2 = B ANGLE = θ DISTANCE = D PERCENTAGE = P

This relationship states that the centroid of a primitive of type A is separated from the centroid of a primitive of type B by a distance of D pixels at angle θ P percent of the time that a primitive of type A is encountered.

To summarize: analysis of edge repetition arrays provided one dimensional texture descriptions. These descriptions are the starting point for a two dimensional texture primitive search. The products of this search are a set of texture primitive masks and descriptions. Using the texture primitive masks a set of placement rules is generated.

If we are working with a homogeneous regular texture, i.e., a periodic texture in which each primitive of a given kind exhibits the same kind of interprimitive spatial relationships, then we should be able to extract a minimum, not necessarily unique, set of rules which completely characterize the underlying texture pattern.

In the event that the texture under consideration is a striped pattern the program would report that each primitive type repeats in the same direction and forms part of the background. Upon detecting this condition the program would only have to select one placement rule for each type of primitive. The rules would relate each primitive type to a given primitive type, say A, in the direction of scan which is precisely the direction of repetition for each primitive type detected.

The remainder of this report describes the algorithm developed to handle homogeneous regular textures in which there is more than one direction of repetition.

Suppose we are given such a texture. If all of the primitives of one type, say type A , are extracted a single primitive "subtexture" of the original texture can by formed. Choosing the 2 nearest

neighbors of a type A primitive so that the 3 primitives are not collinear, we have a configuration similar to the one in Figure 1a. This is equivalemt to choosing 2 noncollinear placement rules relating primitive type A to primitive type A which minimize distance. Since each element of a given type has the same environment the structure in Figure 1a can be propagated to form a triangular or parallelogram grid pattern as in Figure 1b, all of whose nodes correspond to type A elements contained in the "subtexture" pattern.

Suppose there exists a primitive of type A in the "subtexture" which is not represented in the grid pattern. Placing this primitive in the grid, but not in the position of a grid node presents a problem. There is no position that this primitive can take where it would be at least as far from every grid node as the maximum distance of the 2 relationships chosen above, and these relationships were chosen so that interprimitive distances would be minimized. This is a contradiction. Hence there is no primitive of type A in the texture which is not included in the grid.

In order to reconstruct the original texture pattern only one placement rule need be used to relate each additional primitive type to the grid. In each case the relationship would be between a primitive type already included in the grid and a type not yet included.

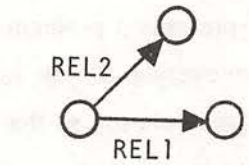
For example, suppose there exists a placement rule relating type A primitives to primitives of type B. This rule could be used to place one type B primitive in the grid for each type A primitive already included, see Figure 1c. All type B primitives placed in this way correspond to a type B primitives in the original texture. Likewise, it can be argued that there exists no type B primitive in the original texture which is not now included in the grid. If there were such a primitive it would not exhibit the interprimitive spatial relationship used to place all of the type B primitives already included in the grid.

Hence, we need only establish the underlying grid pattern using 2 single primitive placement rules, both referring to the same primitive type. Then only one placement rule for each additional non-background primitive type need be used to include the remaining primitive types in this grid structure.

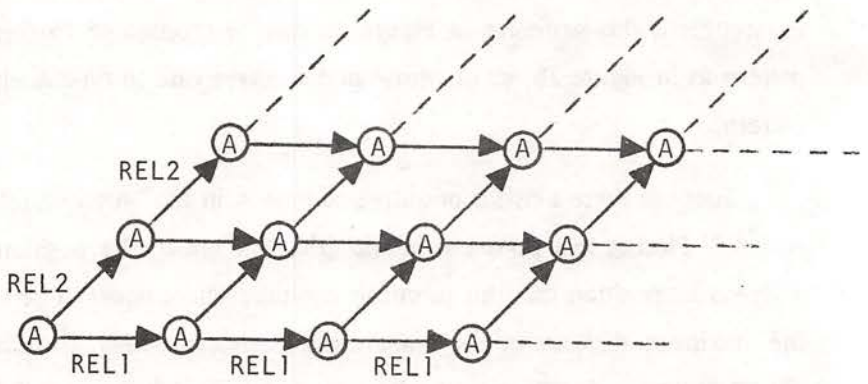
Overview

There are 2 main objectives in this section of the program, namely, selecting a minimum set of grid pattern relations, and reconstructing the original texture pattern using these relations and a set of average primitive templates and descriptions. The latter is done to provide a visual measure of algorithm performance.

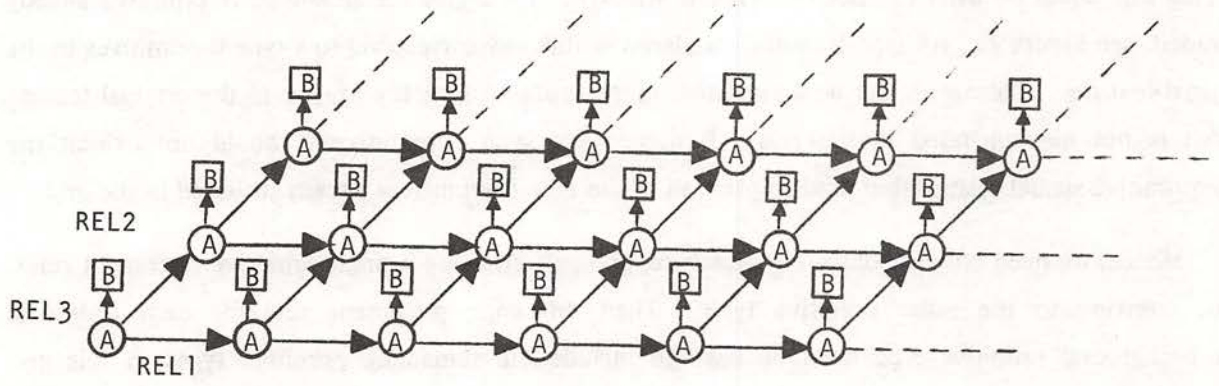
Grid relation selection entails choosing 2 noncollinear single primitive placement rules referring to the same primitive which maximize frequency of occurrence while minimizing interprimitive



(a) 2 Non-collinear single primitive placement rules.



(b) Since each type A primitive has the same environment a parallelogram grid is propagated.



(c) Using an additional relationship, REL3, to add type B primitives to the grid.

Figure 1.

distances. Next, the most frequently occurring placement rules relating elements already included in some grid relation to ones not yet included are choosen until all non - background primitives can be related to the basic grid structure.

Using this set of relations a grid whose nodes represent primitive centroids can be constructed. By replacing each node with the corresponding "average" primitive a pattern similar to the original texture pattern can be created.

Methodology

Grid Relation Selection

1) *Combine redundant single primitive placement rules* - rules relating a primitive to itself at a distance, D, and an angle, θ, are combined with rules relating this same primitive to itself at distance, D, and angle θ + 180 degrees or θ -180 degrees.

2) *Determine Underlying Grid Pattern* - from the reduced set of single primitive placement rules created in 1) choose 2 rules to span the image space as follows:

- Let SET0 be the reduced set of single primitive placement rules.
- Calculate THRESH1 = .85 * (Maximum percentage among all single primitive placement rules in SET0).
- Define SET1 = { Single primitive placement rules from SET0 with percentage greater than THRESH1 }.
- Calculate MIN1 = Minimum distance among all placement rules in SET1.
- Find REL1 in SET1 with maximum percentage among all placement rules in SET1 exhibiting the minimum distance MIN1. REL1 defines the First Grid Axis.
- Redefine SET0 = SET0 - ({ Placement rules not pertaining to the primitive related in REL1 } U { Placement rules whose angle is within 10 degrees of the angle in REL1 or it's opposite }).
- Calculate THRESH2 = .85 * (Maximum percentage among all of the placement rules of SET0).
- Define SET2 = { Single primitive placement rules from SET2 with percentage greater than THRESH2 }.
- Calculate MIN2 = Minimum distance among all placement rules in SET2.
- Find REL2 in SET2 with maximum percentage among all placement rules in SET2 exhibiting the minimum distance MIN2. REL2 defines the Second Grid Axis.

3) *Relate Remaining Primitives to Grid* - all non - background primitives which are not specified in REL1 or REL2 are related to the underlying grid pattern as follows:

- Determine which placement rules are STRONG. Let P1 be a placement rule relating a primitive of type A to a primitive of type B at angle θ and distance D. P1 is STRONG if there exists a placement rule P2 relating a primitive of type B to a primitive of type A at distance D and angle θ + 180 degrees or θ - 180 degrees.
- Redefine SET0 to be the set of all placement rules relating a primitive included in the grid to one which is not yet included.
- Calculate THRESH3 = .85 * (Maximum percentage among all placement rules in SET0).
- Define SET3 = { Placement rules from SET0 with percentage greater than THRESH3 }.
- Determine whether any of the placement rules in SET3 are STRONG , and if so add the highest percentage placement rule which is also STRONG to the set of grid relationships. If no placement rule in SET3 is STRONG then the placement rule exhibiting the maximum percentage is added to the grid relationship set.
- If there are primitives which are still not related to the grid structure then repeat the above 4 steps starting with the redefinition of SET0.

Reconstruct Texture Pattern

1). *Create Primitive Templates* - an "average" primitive mask is created for each non-background primitive type as follows:

- Using a blank image superimpose each primitive of a certain type such that their centers of mass lie on the same pixel.
- Let each pixel value equal the number of primitives covering that pixel.
- Set to zero all pixels with values less than the total number of primitives of that type multiplied by a user specified threshold value.
- After thresholding, the reduced "average" primitive mask is moved to a free section of the template image, and the starting row and column, the location of the center of mass, and the maximum length and width are recorded.

2) *Construct Grid Skeleton* - using the grid relation set mark primitive centroid locations with primitive type numbers as follows:

- Starting at the center of the image mark the primitive centroid locations for one axis using the angle and distance information in the First Grid Axis placement rule. (See Figure 2a.)
- Construct an axis of centroids through each of the First Axis centroid locations using the angle and distance information in the Second Grid Axis placement rule. (See Figure 2b.)
- The rest of the centroids are marked in a similar manner. Using each of the remaining placement rules from the grid relation set produced in I, mark the centroid locations of each "new" primitive type. That is, use the placement rule information to relate a primitive of one type to a primitive of another type whose centroids have already been marked in the image.

0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

(a) First grid axis centroids for placement
Rule: PRIM1 = 1 PRIM2 = 1 ANGLE = 0° DISTANCE = 4.

0	0	0	0	1	0	0	0	1
0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	0

(b) Second grid axis centroids for placement
Rule: PRIM1 = 1 PRIM 2= 1 ANGLE = 45° DISTANCE = $2\sqrt{2}$.

Figure 2.

3) *Fill in Primitives and Background* - using the "average" primitive templates created in 1) and the average primitive intensities calculated earlier in the program the texture pattern can be reconstructed. The reconstructed texture image will have an "average" primitive region centered at each of the primitive centroid markers. The intensity of this region will be equal to the average primitive intensity of the respective primitive type. All pixels not covered by a primitive will take on the background intensity if a background has been detected.

Results

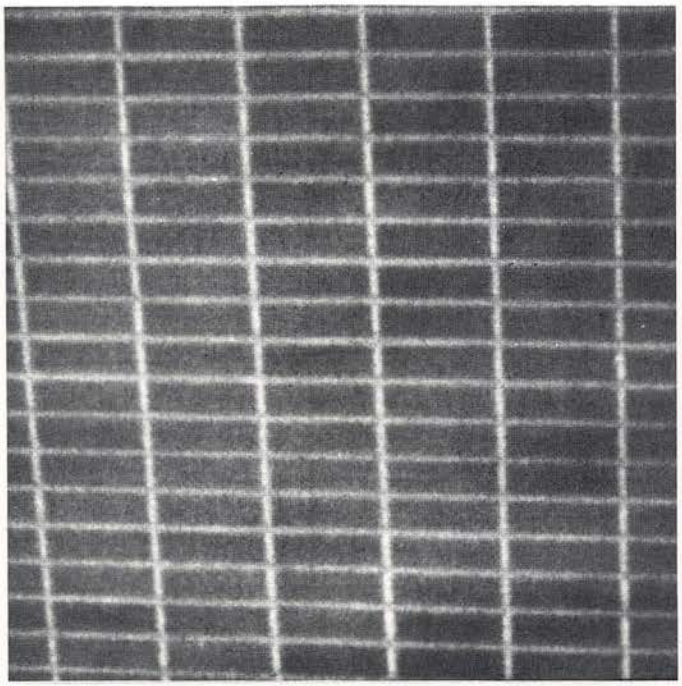
In the previous USCIPI Semiannual Report [1] 2 brick patterns were analyzed, and manual reconstructions of the original texture patterns were attempted. In the results given below grid relations are automatically chosen from sets of placement rules and reconstructions of the original texture patterns are automatically generated. The original brick pattern texture images are given in Figure 3, the texture primitive descriptions are given in Figure 4, and the actual primitives extracted are seen in the composite primitive images of Figure 5.

The set of placement rules are given in Figure 6a for Brick Pattern 1 and Figure 6b for Brick Pattern 2. In each case the single starred entries represent the First Grid Axis placement rules, the double starred entries represent the Second Grid Axis placement rules, and the triple starred entries represent the rules used to link the primitives of type 1 to the underlying grid pattern.

Figure 7 shows the reconstructed texture patterns for Brick Pattern 1 and Brick Pattern 2. In the first pattern, Brick Pattern 1, the vertical mortar strips were determined to form part of the background, see the description of primitive 3 in Figure 4a. Hence all pixels in Figure 7a appear to have a non - zero intensity. (The background primitive, primitive 3, is not displayed in the composite primitive image in Figure 5a). In the second pattern, Brick Pattern 2, no background was detected, also the vertical mortar strips were sometimes not found, sometimes determined to be separate primitives and sometimes they were absorbed by neighboring bricks. Since no background was found and the vertical strips were not indicated as separate primitives by the analysis there are black spaces where these strips should occur. The bricks and horizontal mortar strips, however, are reasonably well placed.

Conclusions

The final descriptions and average primitive masks produced by this method provide resonably complete texture descriptions for homogeneous regular textures. It is hoped that this method can be extended to include more complex texture patterns, and thus aid in the recognition of a larger class of textures.



(a) Brick pattern 1 subwindow.

(b) Brick pattern 2 subwindow.

Figure 3.

```
PRIMITIVE ANALYSIS FOR BRICK 1

RELATIVE INTENSITY IS  LIGHT  DIRECTION IS VERTICAL
NUMBER OF SAMPLES:  87     PRIMITIVE NUMBER IS:  1
  AVERAGE PRIMITIVE DIMENSIONS ARE:  (2.00 AND 52.92)
  AVERAGE PRIMITIVE SIZE IN PIXELS IS:  (105.39)
  AVERAGE PRIMITIVE INTENSITY IS:  (120.80)
  PRIMITIVES REPEAT AT ELEMENT SPACE: (15.00)IN ABOVE MENTIONED DIRECTION

RELATIVE INTENSITY IS  DARK  DIRECTION IS VERTICAL
NUMBER OF SAMPLES:  106     PRIMITIVE NUMBER IS:  2
  AVERAGE PRIMITIVE DIMENSIONS ARE:  (10.00 AND 35.34)
  AVERAGE PRIMITIVE SIZE IN PIXELS IS:  (353.14)
  AVERAGE PRIMITIVE INTENSITY IS:  (100.59)
  PRIMITIVES REPEAT AT ELEMENT SPACING:  (15.00)IN ABOVE MENTIONED DIRECTION

RELATIVE INTENSITY IS  LIGHT  DIRECTION IS HORIZONTAL
NUMBER OF SAMPLES:  8     PRIMITIVE NUMBER IS:  3
  THIS PRIMITIVE FORMS PART OF BACKGROUND
  AVERAGE PRIMITIVE SIZE IN PIXELS IS:  (441.88)
  AVERAGE PRIMITIVE INTENSITY IS:  (132.38)
  PRIMITIVES REPEAT AT ELEMENT SPACING:  (43.00)IN ABOVE MENTIONED DIRECTION
```

Figure 4(a) Brick pattern 1 primitive texture element description.

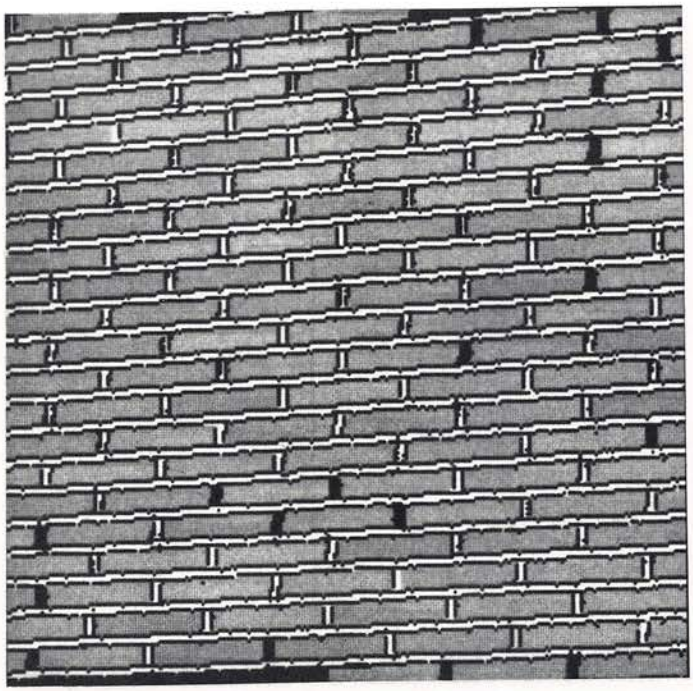
```
PRIMITIVE ANALYSIS FOR BRICK 2

RELATIVE INTENSITY IS  LIGHT  DIRECTION IS VERTICAL
NUMBER OF SAMPLES:  39     PRIMITIVE NUMBER IS:  1
  AVERAGE PRIMITIVE DIMENSIONS ARE:  (2.00 AND 145.26)
  AVERAGE PRIMITIVE SIZE IN PIXELS IS:  (289.87)
  AVERAGE PRIMITIVE INTENSITY IS:  (175.44)
  PRIMITIVES REPEAT AT ELEMENT SPACING: (12.00)IN ABOVE MENTIONED DIRECTION

RELATIVE INTENSITY IS  DARK  DIRECTION IS VERTICAL
NUMBER OF SAMPLES:  122     PRIMITIVE NUMBER IS:  2
  AVERAGE PRIMITIVE DIMENSIONS ARE:  (8.00 AND 40.93)
  AVERAGE PRIMITIVE SIZE IN PIXELS IS:  (326.57)
  AVERAGE PRIMITIVE INTENSITY IS:  (134.25)
  PRIMITIVES REPEAT AT ELEMENT SPACING: (12.00)IN ABOVE MENTIONED DIRECTION
```

Figure 4(b) Brick pattern 2 primitive texture element description.

(a) Brick pattern 1 composite primitives



(b) Brick pattern 2 composite primitives

Figure 5

BRICK1 PRIMITIVE PLACEMENT RESULTS

PRIM1	PRIM2	ANGLE[1] (DEGREES)	DISTANCE (PIXELS)	TOTAL (%)
1	1	-90	15	52.91
1	1	90	15	52.90
1	2	-90	6	***62.72
1	2	90	6	52.25
2	1	-90	6	42.88
2	1	90	6	51.48
2	2	180	45	**65.10
2	2	-90	15	*94.55
2	2	0	45	**63.12
2	2	90	15	*94.55

[1] Negative angles are counter-clockwise; positive angles are clockwise.

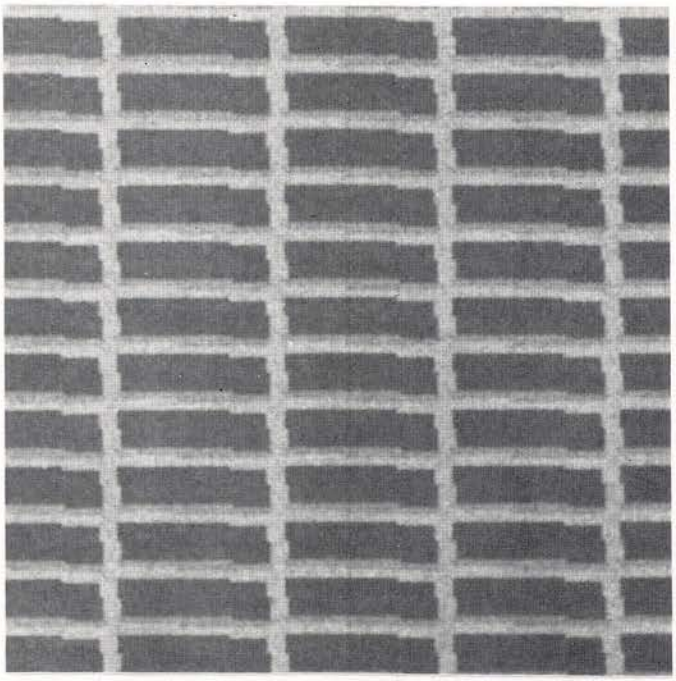
Figure 6(a) Brick pattern 1 interprimitive angle in degrees, distance in pixels and frequency of occurrence.

BRICK2 PRIMITIVE PLACEMENT RESULTS

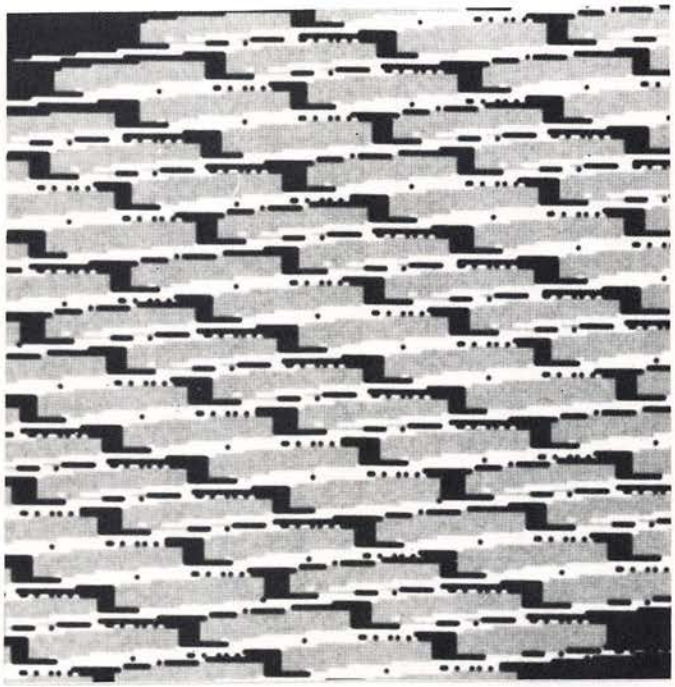
PRIM1	PRIM2	ANGLE[1] (DEGREES)	DISTANCE (PIXELS)	TOTAL (%)
1	1	90	12	18.28
1	1	-90	12	18.27
1	2	-170	18	18.47
1	2	-170	30	18.88
1	2	-30	12	***28.70
1	2	10	24	23.81
1	2	10	33	18.76
1	2	80	6	20.70
2	2	-160	24	*36.63
2	2	-90	12	19.19
2	2	-30	27	**33.17
2	2	0	45	36.53
2	2	20	24	*36.41
2	2	90	12	20.86
2	2	150	27	**32.52
2	2	180	45	36.18

[1]Negative angles are counter-clockwise; positive angles are clockwise.

Figure 6(b) Brick pattern 2 interprimitive angle in degrees, distance in pixels and frequency of occurrence.



(a) Brick pattern 1 reconstruction.



(b) Brick pattern 2 reconstruction.

Figure 7.

References

[1] F. Vilnrotter, R. Nevatia and K.E. Price, "Determining Spatial Relationships Between Texture Primitives in Homogeneous Regular Textures," USCIPI Semiannual Technical Report 990 September 1980, pp. 10-26.

[2] L.S. Davis, "Computing the Spatial Structure of Cellular Textures," *CGIP-11*, pp. 111-122, 1979.

[3] J.T. Maleson, "Understanding Texture in Natural Images," University of Rochester, Ph.D. Thesis, to be published.

[4] T. Matsuyama, K. Saburi and M. Nagao, "A Structural Description of Regularly Arranged Textures," *5-IJCPR*, Miami, Fl., Dec. 1980, pp. 1115-1118.

[5] R. Nevatia, K. Price and F. Vilnrotter, "Describing Natural Textures," USCIPI Semiannual Technical Report #860 March 1979, pp. 29-54.

[6] F. Vilnrotter, R. Nevatia and K. Price, "Automatic Generation of Natural Texture Descriptions," USCIPI Semiannual Technical Report #910, September 1979, pp. 31-63.

[7] F. Vilnrotter, R. Nevatia, and K. Price, "Extraction of Texture Primitives," USCIPI Semiannual Technical Report #960, March 1980, pp. 48-59.

[8] R. Nevatia, K. Price, and F. Vilnrotter, "Describing Natural Textures," *Proc. of the Sixth IJCAI-79*, Tokyo, Japan, August 1979.

[9] F. Vilnrotter, R. Nevatia, and K. Price, "Structural Description of Natural Textures," *Proc. of the Fifth IJCPR-80*, Miami, Florida, December 1980.

1.6. Texture Identification and Segmentation

David D. Garber **and** *Alexander A. Sawchuk*

Introduction

In this paper we examine various approaches to texture segmentation and identification using the linear model developed in earlier papers [1,2,3]. These methods employ covariance measures of a texture over windows of the region being segmented or identified. These same covariance measures were used to estimate the linear model parameters which were examined in earlier texture synthesis models [1]. The information content of these measures was shown by performing simulations of various textures and therefore, based on these results, we might conclude that these second-order statistics would be useful in the segmentation and identification of textures. Pictorial segmentation results are given in this paper.

It is generally agreed that a great portion of texture information is contained in the second-order statistics of a texture. There are notable exceptions to this rule as was shown by Gagalowicz [6], however for most natural textures, second-order statistics have proved to be sufficient to adequately discriminate textures in most applications [4,5]. In a practical sense, we usually are also prevented from gathering and analyzing higher-order statistics because of computational limitations. In fact, we can easily be overwhelmed by masses of data arising from second-order statistics.

The most general second-order statistics are the complete second-order joint densities. These measures may be estimated by joint gray scale histograms taken over a parent texture. For a g grey-level image, each histogram requires g^2 storage locations. But, there is one such histogram for each vector distance or spacing between a pair of pixels. To compute these histograms over all spacings, $2,...,n_r$, in two dimensions would result in $(n_r-1)^2g^2$ entries. Even for reasonable g and n_r, this could easily create a data expansion containing unnecessary information rather than a data reduction yielding measurements with discrimination value.

For these reasons, texture image data is often quantized (to reduce g) and second-order measurements resulting in joint grey-scale histograms are made over a small number of pixel spacings (to reduce n_r). To decrease the size of the feature space further, various functions of joint grey-scale histograms are calculated and these values are primarily used to identify a texture. For a single histogram, as many as 25 to 30 functions have been proposed [7]. In spite of the large dimensionality of the feature space and the problems with quantizing low contrast textures, this family of texture features is used frequently to successfully classify textures [7,8].

Many other identification and classification schemes exist [9] as the discrimination of textures represents the most important application of texture analysis.

In this paper, we reduce the information contained in the joint grey-scale histogram to one single number, the correlation coefficient for that particular pixel spacing. It is expected that this large reduction will cause a decrease in discrimination power as the size and information content of the feature space has been significantly reduced. The purpose of this exercise is not to develop a new, more powerful texture identifier but merely to access the information content of the correlation coefficient values when applied to the problem of texture discrimination. It is already apparent from the simulation results presented in earlier papers that a great deal of texture information may be obtained by proper use of these correlation coefficients.

Correlation measurements have been applied in various ways to the problem of texture identification previous to this study [4,8-14]. It has generally been concluded that they are of lesser value than Haralick's family of functions on values of the joint grey-scale histogram [5] when applied to the discrimination problem. In this paper we will develop two new discrimination methods utilizing correlation values. One is based on the statistical test for equality of covariance matrices. The other utilizes multiple statistical tests for the equality of individual correlation coefficients. Both show good discrimination power but neither exceeds the quality of Haralick's measures.

Segmentation Using Correlation Matrices

Texture is a feature which can only be measured and identified over an area of an image. Therefore most segmentations of an image according to texture information will require that measurements be taken over an area and then part or all of that area will be classified accordingly. In our work, measurements were taken over a square, W_{BIG} pixels in length, and a center square, W_{SMALL} pixels in length, was classified from these measurements as in Fig. 1.

Second-order statistics must be measured over a variety of vector distances (pixel-pair spacings) to be useful in texture discrimination. In our case, these second-order statistics are correlation values. There is only one correlation value for a particular spacing.

There are many approaches to estimating a correlation value over a window for a particular pixel-pair spacing. One involves passing a kernel of pixels over the window and taking a sample at all points where the kernel is completely contained within the window boundaries. Such measurements would result in a covariance matrix for the kernel over that window. This matrix can be used to identify a texture, as will be shown later in this section. Another approach to estimating a correlation value for a particular pixel-pair spacing would involve measurements over all possible samples within the window of

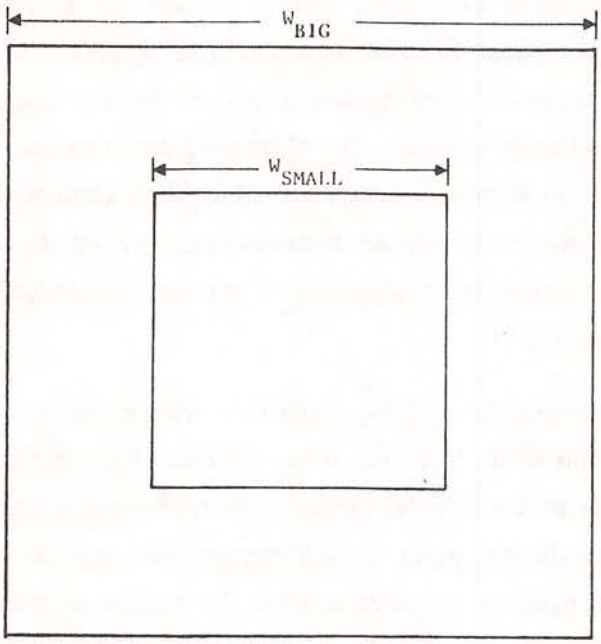


Fig. 1. Segmentation window.

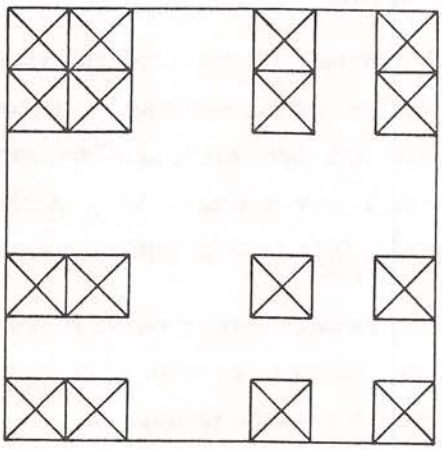


Fig. 2. Two-dimensional spanning kernel.

that spacing. This is equivalent to passing a kernel containing two points over the window for each pixel-pair spacing. The result of this approach is a correlation value for each spacing which must be examined by itself. A method for identifying texture using these individual correlation values will be examined in section 3.

Once we have obtained a covariance matrix by passing a kernel of points over a window, that matrix may be compared statistically with covariance matrices from known textures. Such a statistical test has been derived for testing the equality of covariance matrices. The maximum likelihood ratio approach used to derive this test makes the standard assumptions of multi-dimensional normality and independence of samples. Both were used in our texture simulation work. Details concerning the test and its derivation are given in [15,16]. The statistical test for two covariance matrices is given by

$$U_1 = 2.3026\ \mathrm{dB} \cong \chi^2_{N(N+1)/2}$$

where

$$D = (M_1+M_2-2)\log_{10}|C| - (M_1-1)\log_{10}|C_1| - (M_2-1)\log_{10}|C_2|$$

C_1 and C_2 are the estimated covariance matrices for each group,

$$C = [(M_1-1)C_1+(M_2-1)C_2]/(M_1+M_2-2)$$

and

$$d = 1 - \left[\frac{1}{(M_1-1)}+\frac{1}{(M_2-1)} - \frac{1}{(M_1+M_2-2)}\right]\left[(2N^2-3N-1)/6(N+1)\right]$$

N is the size of the covariance matrices being tested which is equal to the number of pixels in the kernel. M_1 and M_2 are the sample sizes used to estimate C_1 and C_2. $|C|$ denotes the determinant of C. The test statistic, U_1, has an approximate chi-square distribution with $N(N+1)/2$ degrees of freedom and approaches 0 as C_1 approaches C_2.

Having derived the test for equality of two covariance matrices we must determine the contents of these matrices. As the number of points in the kernel increases, the size of the covariance matrices increases leading to more difficult and time-consuming determinant calculation required in Eq. 1. Also, as the spacing of these pixels increases, the number of samples in a window decreases. Finally, as more and more points are included in the kernel, the amount of redundant and overlapping information in the covariance matrix increases due to redundant pixel-pair spacing.

To eliminate this redundancy, we will consider the problem briefly in one-dimension. Perfect, non-redundant pixel spacing is possible only for patterns with maximum range of 1, 3 or 6 pixels given by the corresponding patterns XX, XX-X and XX--X-X as shown in Table 1. For these three particular

Table 1. MINIMAL SPANNING SETS.

RANGE	#PTS	
2	2	XX
3	3	XXX
4	3	XX-X
5	4	XXX-X
6	4	XXX--X
7	4	XX--X-X
8	5	XXXX---X
9	5	XXX--X--X
10	5	XXX---X--X
11	6	XXXX--X---X
12	6	XXXX---X---X
13	6	XXXX----X---X
14	6	XXX---X---X--X
15	7	XXXXX----X----X
16	7	XXXXX-----X----X
17	7	XXXX----X---X---X
18	7	XXXX----X----X---X
19	8	XXXXXX------X-----X
20	8	XXXXX----X----X----X
21	8	XXXXX-----X----X----X
22	8	XXXXX-----X-----X----X
23	8	XXXX----X----X----X---X
24	8	XXX--------X---X--X--X-X
25	9	XXXXXX------X-----X-----X
26	9	XXXXXX------X------X-----X
27	9	XXXXX-----X----X-----X----X
28	9	XXXXX-----X-----X-----X----X
29	9	XXX----------X----X--X--X--XX
30	9	XXX----------X---X--X--X--X-X
31	10	XXXXXX-----X------X-----X-----X
32	10	XXXXXX------X-----X------X-----X
33	10	XXXXXX------X------X------X-----X
34	10	XXXXX-----X-----X-----X-----X----X
35	10	XXXX----------X----X---X---X--X--X
36	10	XXX-------------X---X--X--X--X--X-X
37	10	XX-X--X------X------X------X---X---XX
38	11	XXXXXXX------X-------X-------X------X
39	11	XXXXXX------X------X-----X------X-----X
40	11	XXXXXX------X------X------X------X-----X
41	11	XXXXX-----X------X------X----X-----X----X
42	11	XXXX----------X----X------X----X---X---X-X
43	11	XXXX-------------X----X---X---X---X--X--X
44	11	XX-X--X------X------X------X------X---X---XX
45	12	XXXXXXX-------X-------X------X-------X------X
46	12	XXXXXXX-------X-------X-------X-------X------X
47	12	XXXXXX------X------X------X------X------X-----X
48	12	XXXXXX------X-------X-------X-----X------X-----X
49	12	XXXX---------X-----------X--X---X-----X---X---XX
50	12	XXXX-----------------X----XX---X---X---X---X--X
51	12	XXXX-----------------X----X---X---X---X---X--X--X
52	13	XXXXXXXX--------X--------X--------X--------X-------X
53	13	XXXXXXX-------X------X--------X------X-------X------X
54	13	XXXXXXX-------X-------X-------X-------X-------X------X
55	13	XXXXXXX-------X--------X--------X------X-------X------X
56	13	XXXXXX------X-------X-------X-------X-----X------X-----X

ranges, the patterns shown are constructed so that no two pairs of X's are separated by the same distance. One and only one pair of X's may be found in these patterns which is separated by each distance less than or equal to the maximum range. For all other ranges, redundancies will exist in any pattern spanning the range. That is, more than one pair of X's may be found which are separated by the same distance in patterns not having a maximum range of 1, 3 or 6. A pattern spans a range if and only if at least two X's may be found in the pattern which are separated by every distance less than or equal to the maximum range. These patterns are sometimes referred to as difference sets [17,18,19].

A set containing a minimal number of X's can be found to span all ranges. A list containing the minimum number of X's required to span each range and one non-unique pattern which spans the range is given in Table 1. Extending these one-dimensional spacings to two dimensions merely requires the vector product of the transpose of any of the row vector patterns with itself yielding a two dimensional matrix.

A two-dimensional redundant kernel containing 16 pixels which spans a two-dimensional range of 6 is shown in Fig. 2. Passing this kernel over a window yields a covariance matrix of dimension 16, which is not an unreasonable size for computation purposes. The number of data samples over a window of size W_{BIG} is $(W_{BIG}-6)^2$. Even for small windows of size 16, a reasonable sample size of 100 may be obtained.

Once the kernel and procedure are determined, we proceed with the process of segmenting the textures to test the identification procedure. There are many approaches to this problem which are usually defined by the particular case of interest. We may or may not have prototype and parent texture data. We can segment, cluster-analyze or identify. Notice that the generality of the test leaves a wide variety of options open. We can compare matrices in one area of an image with those in other areas and our test statistic values will be the distance measures defining the closeness of the textured regions. We could compare a matrix with a number of known prototypes and classify the unknown region according to the test statistic for each comparison. Finally, we may merely wish to locate a particular texture in an image and thus we would compare matrices with only one prototype.

This later approach was selected in this paper as the visual results are simple to display and analyze. We will compare the matrix measured over a large window with one obtained over the complete parent texture prototype and assign the pixels in the small window accordingly (see Fig. 1). Figure 3 shows the test composite image used. We will attempt to identify the texture sand in that region. The texture sand was chosen as the simulation results of this texture were in some ways very poor when examined critically on a high resolution device. Therefore, it represents a worst-case example.

Figure 4 shows the segmentation results when $W_{BIG} = W_{SMALL} = 16$. The pixel brightness are linearly mapped test statistic values, which are supposed to be chi-square-distributed. Improvement is made when $W_{BIG} = 32$ and $W_{SMALL} = 16$ as is seen in Fig. 5. Figure 6 shows much improved results when $W_{BIG} = 48$ and $W_{SMALL} = 16$. The difference is due to the availability of more texture information in the larger window in the form of increased sample size (100 to 1764).

As a final note, it should be understood that this chosen texture kernel (see Figure 2) totally ignores information obtained earlier concerning the interdependence of pixels in the generating kernel as expressed in the linear model. It is very possible that better segmentation results could be obtained by using the kernel shape which best fits each texture as this particular pattern of points was found to be most significant. The linear model used for simulation of each texture might even be used itself as the distributions of the parameter estimates are known if certain assumptions are made [20,21]. But neither of these approaches have been tried even though improvements are expected.

Segmentation Using Individual Correlation Coefficients

Individual correlation estimates for particular pixel-pair spacings may be made by taking measurements over all possible samples within the window containing that spacing. This approach utilizes more information than the fixed kernel method as it includes measurements near all edges of the window (thus, the sample size is increased). The result is a single coefficient for each pixel-pair spacing. If the spacing between each pair is unique then the set of coefficients will be non-redundant. The covariance matrices used to segment textures in the last section may contain redundant information if the pixel-pair spacings in the kernel are repeated.

A statistical test for comparison of two correlation coefficients has been derived for the bivariate normal distribution and is

$$U_2 \doteq \chi_1^2$$

where

$$U_2 = (Z_1-\bar{Z})^2(M_1-3)+(Z_2-\bar{Z})^2(M_2-3)$$

and

$$Z_1 = \operatorname{arctanh}(r_1)$$

$$Z_2 = \operatorname{arctanh}(r_2)$$

$$Z_3 = [(M_1-3)Z_1+(M_2-3)Z_2]/(M_1+M_2-6)$$

r_1 and r_2 are the computed correlation coefficients from the two groups. This test is derived using the

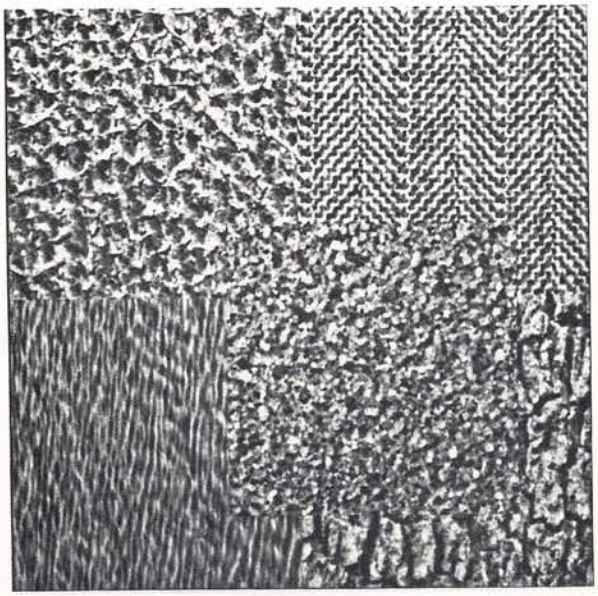


Fig. 3. Input composite image.

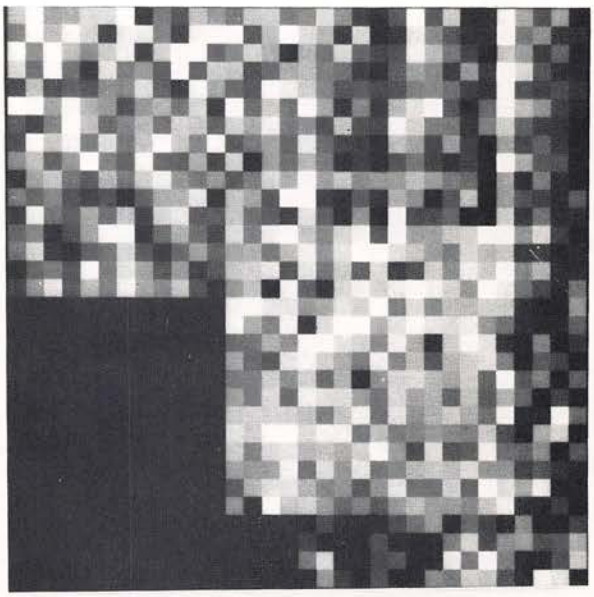


Fig. 4. Segmentation using covariance matrix $W_{BIG} = 16$.

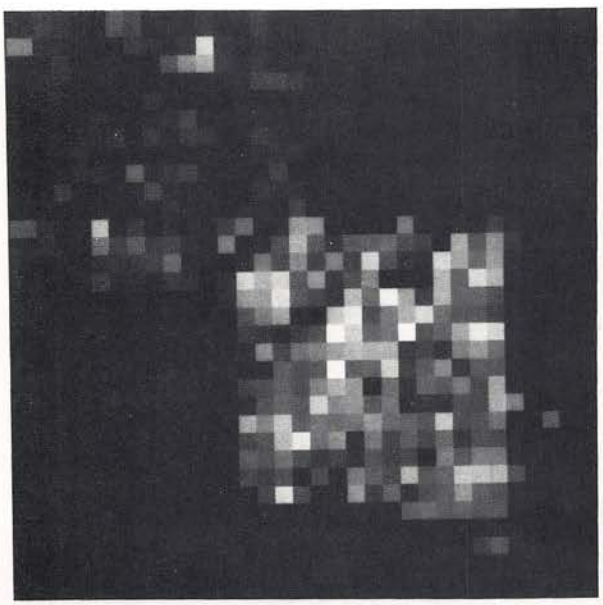


Fig. 5. Segmentation using covariance matrix $W_{BIG} = 32$.

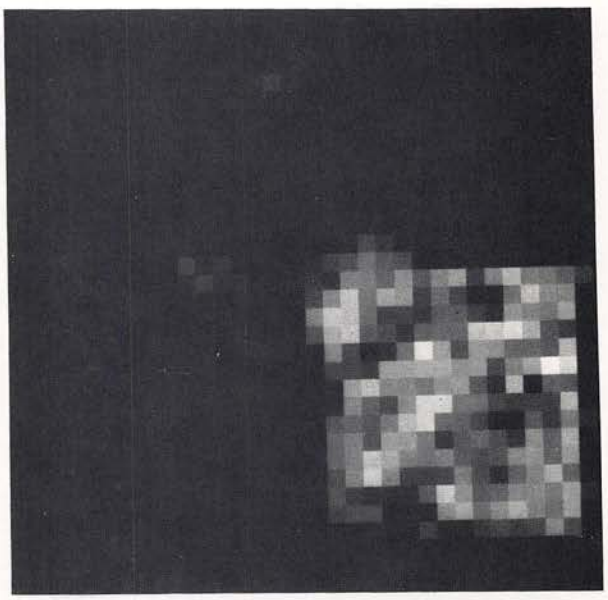


Fig. 6. Segmentation using covariance matrix $W_{BIG} = 48$.

fact that Z_1 is approximately distributed $N(Z_1: \text{arctanh}(\rho), (M_1\text{-}3)^{-1})$ where ρ_1 is the true correlation coefficient of the population. For additional details see [21]. Again, to derive the tests assumptions of normality and independent samples are made.

As in the previous section, the test statistics may be used to segment or identify textures. However, difficulty arises as our test must be performed for each measured correlation value. Thus a number of U_2 values are obtained if multiple pixel-pair spacings are used. At this point these values can be combined in numerous ways. As each U_2 value is chi-square distributed we can compute the probability, p_{u_2}, that a random variable, which has that chi-square distribution, is greater than or equal to U_2 as shown in Fig. 7. For $U_2 = 0$ ($r_1 = r_2$) this probability is 1.0. A product of the probabilities associated with each U_2 was used to indicate the overall fit of one set of correlation coefficients to another set.

Finally, it should be noted that tests for equality of correlation coefficients will not detect differences in mean and variance over the window. The correlation coefficient specifically removes these effects. As a result, it may be advantageous to include tests for equality of means and variances into the comparison process. For equality of means the test statistic is

$$t_1 = \frac{\hat{\mu}_1 - \hat{\mu}_2}{\sqrt{\frac{(M_1-1)\hat{\sigma}_2^2 + (M_2-1)\hat{\sigma}_2^2}{M_1+M_2-2}}\sqrt{\frac{1}{M_1}+\frac{1}{M_2}}} \cong t_{M_1+M_2-2}$$

where $\hat{\mu}_1$ and $\hat{\mu}_2$ are the calculated mean and $\hat{\sigma}_1^2$ and $\hat{\sigma}_2^2$ are the calculated variances for the two groups being compared. This test statistic is a value of a random variable having the t-distribution with $M_1+M_2\text{-}2$ degrees of freedom. For the equality of variances the test statistic is

$$f_1 = \frac{\hat{\sigma}_1^2}{\hat{\sigma}_2^2} \cong F_{M_1-1,M_2-1}$$

This test statistic is a value of random variable having an F-distribution with $M_1\text{-}1$ and $M_2\text{-}1$ degrees of freedom. The derivation of both of these tests requires the assumption of normality for the two groups and an independent sample set. The known distributions of the test statistics t_1 and f_1 was used to obtain the probabilities, p_{t_1} and p_{f_1} that a random variable having that distribution would be greater than or equal to the calculated values of $|t_1|$ and $\text{MAX}(f_1, 1/f_1)$ or less than $-|t_1|$ and $\text{MIN}(f_1, 1/f_1)$ (see Figs. 8 and 9).

Having obtained the set of probabilities, p_{u_2}, and p_{t_1} and p_{f_1} corresponding to the set of tests for the equality of correlation coefficients and the tests for equality of mean and variance of the window a

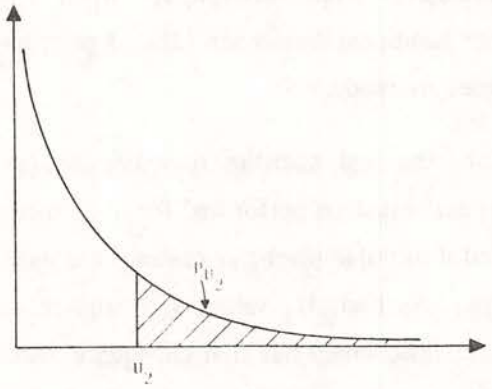


Fig. 7. The chi-square distribution.

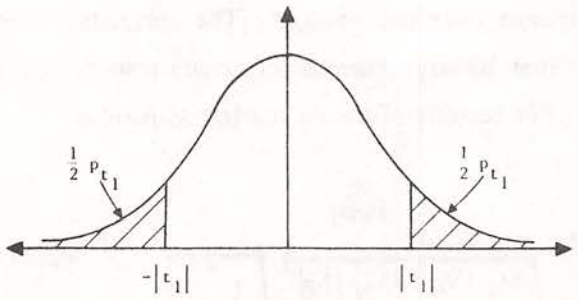


Fig. 8. Student's t-distribution.

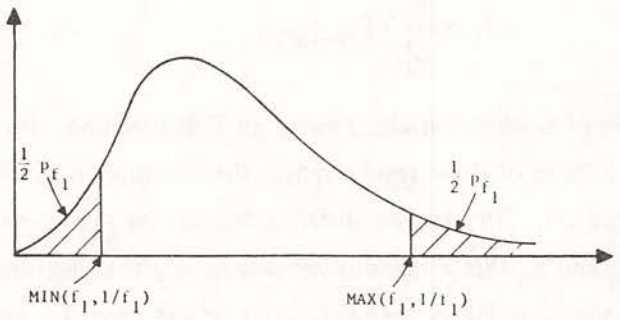


Fig. 9. The F-distribution.

single test statistic

$$\sum_i \log (\text{MAX}(p_{U_{2_i}}, 10^{-6})) + \log p_{t_1} + \log p_{f_1}$$

is formed to indicate the combined fit of these tests. The probabilities are log-scaled to eliminate scaling problems which would occur when taking the product of many small numbers. The $p_{u_{2_i}}$'s are bounded below to reduce the effect of single, noisy correlation coefficients. These output test statistics obtained by comparing a windowed region of Fig. 3 with statistics gathered from the parent prototype texture sand were linearly scaled to produce the intensity images shown in Figs 11 to 13. The results for $W_{BIG} = W_{SMALL} = 16$ are shown in Fig. 11. Improved results for $W_{BIG} = 32$ and $W_{SMALL} = 16$ are shown in Fig. 12. Results for $W_{BIG} = 48$ and $W_{SMALL} = 16$ are shown in Fig. 13.

Conclusions

In implementing the two procedures detailed in this chapter, large values for the test statistics U_1 and U_2 were obtained even when comparing the matrices or correlation values measured from an extracted portion of a parent texture with those obtained from the entire parent texture. There are two basic reasons for this. First, the assumptions of normality are probably incorrect. There is little reason to believe that the distribution of pixels in an image is truly multi-variate normal. Secondly, the samples used to estimate the covariance matrix and the correlation coefficients are not random and independent. They are strongly related by their spatial positions in the image and are often adjacent. The sample size may be adjusted to reflect this fact but no work in this area was done.

In most cases, statistical tests may be considered to be robust - not weakened significantly - when assumptions used to derive them are violated. This is probably not true of the test for equality of two covariance matrices which has been called "frail at best" [15]. For these reasons, the methods presented may be considered innovative but not necessarily statistically sound.

In spite of this fact, the results show that the method using the test for equality of covariance matrices was superior to the method involving multiple tests of individual correlation coefficients. This could be due to the method used to combine the multiple tests.

The pictorial results of this chapter indicate the usefulness of correlation values in texture identification. The methods are not intended to be improvements over existing segmentation identification techniques. (There are enough of them already.) The discussion is intended to apply a specific texture model which was based on second-order statistics to the texture identification problem and this was done successfully.

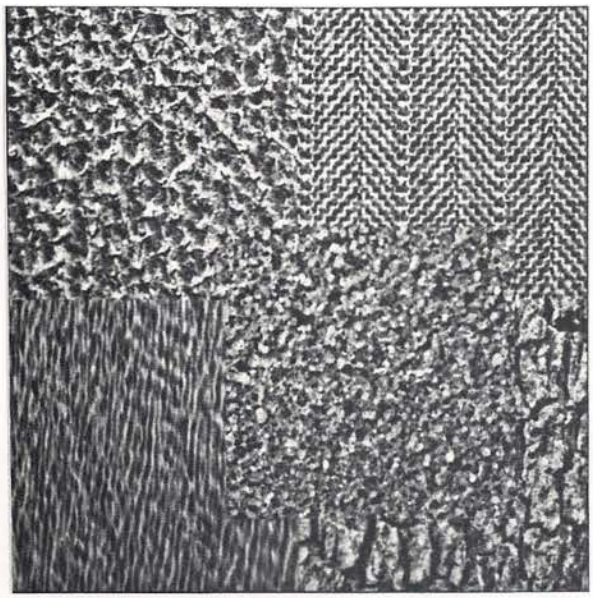


Fig. 10. Input composite image

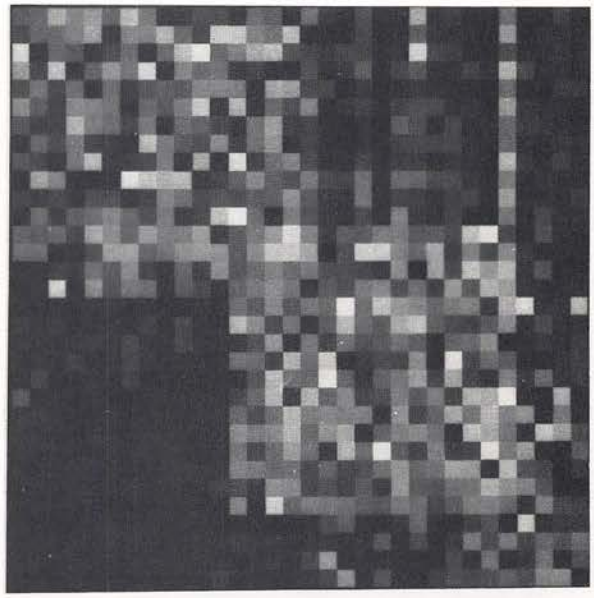


Fig. 11. Segmentation using individual correlation values $W_{BIG} = 16$.

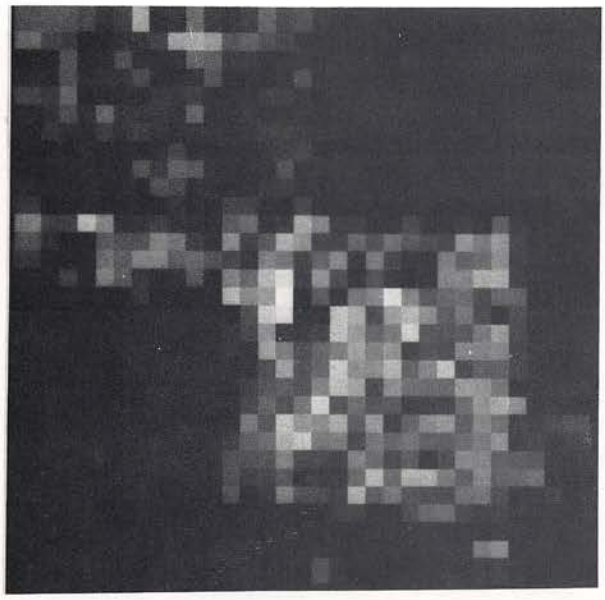


Fig. 12. Segmentation using individual correlation values $W_{BIG} = 32$.

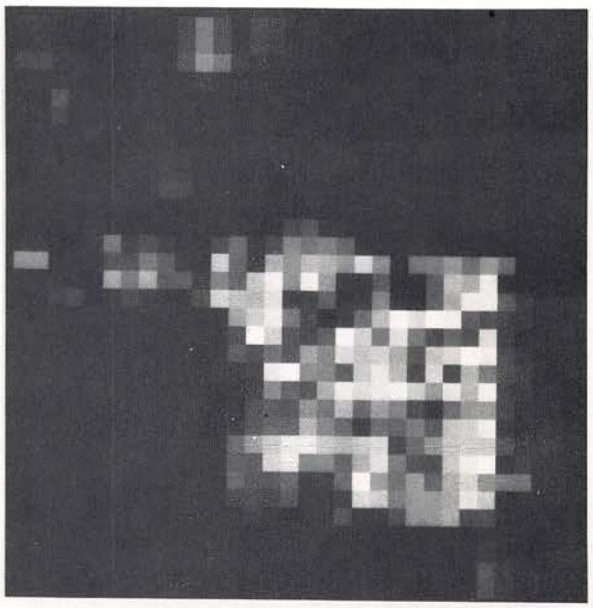


Fig. 13. Segmentation using individual correlation values $W_{BIG} = 48$.

References

[1] D.D. Garber, "Models for Texture Analysis and Synthesis," *USCIPI Report 910*, Image Processing Institute, Univ. of Southern California, Los Angeles, Sept. 1979.

[2] D.D. Garber, "Application of the General Linear Model to Binary Texture Synthesis," *USCIPI Report 960*, Image Processing Institute, Univ. of Southern California, Los Angeles, March 1980.

[3] D.D. Garber, "Higher-Order Texture Synthesis Models and Residual Examination," *USCIPI Report 960*, Image Processing Institute, Univ. of Southerm California, Los Angeles, March 1980.

[4] R.M. Haralick, "Statistical and Structural Approaches to Texture," *Proc. of the IEEE*, vol. 67, no. 5, May 1979.

[5] 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.

[6] A. Gagalowicz, "Visual Discrimination of Stochastic Texture Fields Based Upon Their Second Order Statistics," *Proc. of Fifth Int. Joint Conf. on Pattern Recognition*, Miami Beach, Dec. 1980, pp. 786-788.

[7] R.W. Conners, *Some Theory on Statistical Models for Texture and Its Application to Radiographic Image Processing*, Ph.D. Thesis, College of Engineering, Univ. of Missouri, Columbia, Dec. 1976.

[8] P. Chen and T. Pavlidis, *Segmentation by Texture Using a Coocurrence Matrix and a Split-and-Merge Algorithm.* TR-237, Princeton Univ., Princeton, NJ, Jan. 1978.

[9] B.H. McCormick and S.N. Jayaramamurthy, "Time Series Models for Texture Synthesis," *Int. J. of Computer and Info. Sciences*, vol. 3, pp. 329-343, Dec. 1974.

[10] J.T. Tou, D.B. Kao, and Y.S. Chang, "Pictorial Texture Analysis and Synthesis," *Proc. of Third Int. Joint Conf. on Pattern Recognition*, Coronado, Calif., Nov. 1976, p. 590.

[11] K. Deguchi and I. Morishita, "Texture Characterization and Texture-Based Image Partitioning Using Two-Dimensional Linear Estimation Techniques," *IEEE Trans. on Computers*, vol. C-27, pp. 739-745, Aug. 1978.

[12] W.K. Pratt, O.D. Faugeras, and A. Gagalowicz, "Visual Discrimination of Stochastic Texture Fields," *IEEE Trans. on Syst., Man and Cybern.*, vol. SMC-8, no. 11, pp. 794-804, Nov. 1978.

[13] H. Kaizer, *A Quantification of Textures on Aerial Photographs*, Note 121, AD 69484, Boston Univ. Research Labs., Boston, 1955.

[14] S.R. Purks and W. Richards, "Visual Texture Discrimination Using Random Dot Patterns," *J. Opt. Soc. Am.*, vol. 67, pp. 765-771, June 1977.

[15] M. Kendall and A. Stuart, *The Advanced Theory of Statistics, Volume 3*, London: Griffin, 1976.

[16] C.Y. Kramer, *A First Course in Methods of Multivariate Analysis*, Virginia Polytechnic Institute and State University, Blacksburg, Virginia, 1972.

[17] F.D. Russell, "Non-Redundant Arrays and Post Detection Processing for Aberration Compensation in Incoherent Imaging," *J. Opt. Soc. Am.*, vol. 61, no. 2, Feb. 1971.

[18] J. Leech, "On the Representation of 1,2,...,n by Differences," *Journal of the London Mathematical Society*, vol. 31, part I, pp. 160-169, April 1976.

[19] F.D. Russell, *Predetection and Postdetection Filtering for Improved Resolution in Optical Systems*, Technical Memo 70-007, Philco-Ford Corporation, Palo Alto, California, Aug. 1970.

[20] B. Julesz, E. Gilbert, and J.D. Victor, "Visual Discrimination of Textures with Identical Third order Statistics," *Biological Cybernetics*, vol. 31, pp. 137-140, 1978.

[21] F.A. Graybill, *Theory and Application of the Linear Model*, North Scituate, Massachusetts: Duxbury Press, 1976.

1.7. Feature Selection In Texture Edge Detection

H.Y. Lee

Introduction

A texture edge detector which is patterned after the methods for detecting intensity edges using directional derivatives was developed [1]. A directional derivative is defined as the difference of two texture measures taken over adjacent, nonoverlapping neighborhoods. Combining these directional derivatives, the presence/absence of a texture edge at every pixel is detected.

A set of texture energy measures which was developed by K. Laws [2] is adopted for texture feature extraction. As a method of combining the edge information coming from these different sources, the selection of the "best" frames in terms of some criteria based on the connectivity of edge elements have been studied. The best frames are combined to form a hybrid plane which can be used as the input of the same texture edge/line detector or some other region based image segmentor.

Connection of Texture Edge Points

To form object outlines or boundaries, edge points obtained from some edge detector should be linked together. In other words, short spurious lines and isolated edge points are deleted and small gaps along the outline are bridged by straight line connections under some condition or filled in by expansion of edge points. Thus, to organize edge information, we form a list of edge points that probably lie on the same physical boundary of an object. A scan runs through all edge points in the picture, and two edge points are linked if certain loose conditions are met:

1. x and y coordinates match within 3 pixels,
2. their angles match within 90 degrees.

If an edge point can be linked to another edge point which is already a member of a set of connected edge points (line), it becomes a new member of that line. If two edge points neither of which is a member of existing lines can be linked together, they form a new line. After this edge linking stage is finished, only lines having more points than some prescribed threshold are selected to survive.

Texture edge detection and linking have been applied to the composite image shown in Fig. 1.

The results of texture edge detection and linking performed on one of texture energy planes are shown in Fig. 2 and Fig. 3, respectively.

Connectivity of Edge Points as Means of Feature Selection

Figure 4 shows 15 available texture energy planes obtained from the composite image of Fig. 1. Some of them are obviously inferior to the others or bear no reliable information about texture boundary location. In statistical pattern recognition area, there are several methods of feature selection based on statistics of individual classes that form the total pattern space [4]. But these methods often assume that the class of a given sample is known (supervised) and concentrate on the faithful reconstruction of pattern space after feature reduction. Coleman and Andrews [5] used Bhattacharyya measure as the criterion for feature reduction in their image segmentation scheme. Clustering using all the features was performed prior to feature selection thus enormous computing time was required. Besides, possibly misleading features are participating in class determination, thus may cause unreliable clustering itself.

Therefore, the need for computationally simple but reliable means of feature selection is obvious. On the fact that the boundary of an object in an image is continuous, edge points detected on "good" feature spaces should form smooth, long connected sets of edge points and the number of isolated edge points should be small. Two measures following this idea are suggested:

1. Average number of edge points in the connected sets of edge points (no thresholding is performed before averaging).
2. Ratio of the number of edge points surviving after linking and thresholding to the number of original edge points obtained by the texture edge detector.

These two measures computed for each of 15 texture energy planes are shown in Table 1 and Table 2, respectively. If we select only planes with both measures above average values taken over all 15 features, we can see the reduced set of texture energy planes (7th, 9th, 10th, 11th, and 13th plane) well matches with our psychovisual selection.

The next step is combining this reduced set of feature planes in a meaningful way. Figure 5 shows a hybrid plane which is the weighted sum of "best" planes with weighting factor used just for normalizing. Another possible way is to use the above two connectivity measures as the weighting factors, thus weighting heavily on the plane with high value of connectivity measures. This hybrid plane shows considerable improvement in homogeneities of different regions and more clear boundaries when we compare the hybrid plane with the 10th texture energy plane which has the higher values in both connectivity measures. Figure 6 shows the result of texture edge detection followed by linking and thresholding when the hybrid plane was used as input. For the comparison purpose, the results of

region-based segmentation (simple thresholding) performed on the hybrid plane and 10th energy plane are shown in Fig. 7.

References

[1] O.D. Faugeras and H.Y. Lee, "Texture Edge Detection," Technical Report USCIPI 960, 1980.

[2] K.I. Laws, "Texture Image Segmentation," Department of Electrical Engineering, Ph.D. Dissertation, January 1980.

[3] K. Fukunaga, Introduction to Statistical Pattern Recognition, Academic Press, New York, 1972.

[4] G.B. Coleman and H.C. Andrews, "Image Segmentation by Clustering," *Proceedings of the IEEE*, pp. 773-785, May 1979.

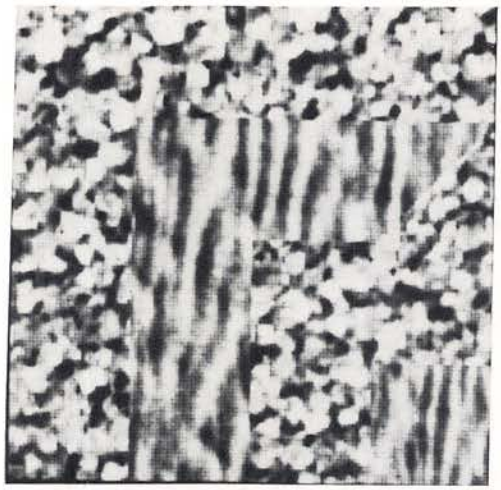


Fig. 1. Composite image of water and sand.

Fig. 2. Texture edges detected on 10*th* plane.

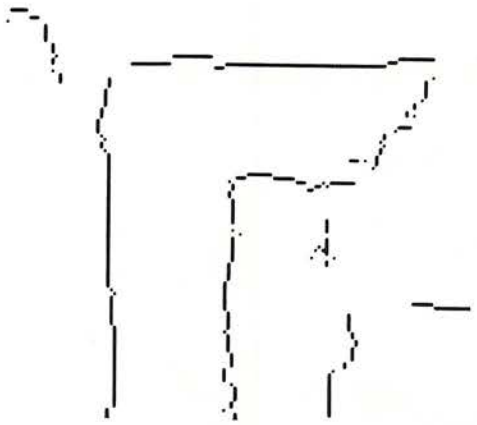
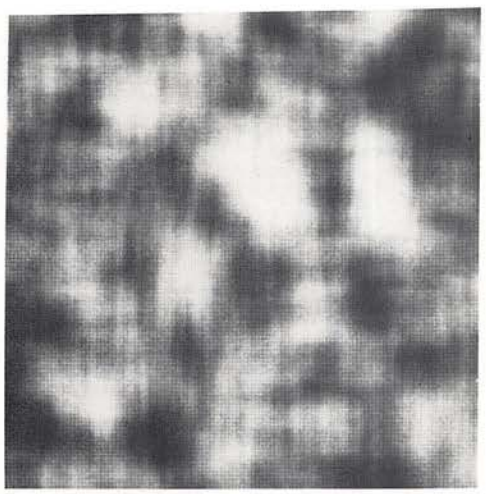
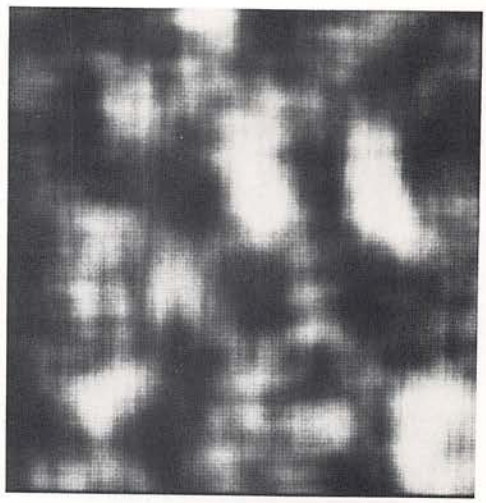


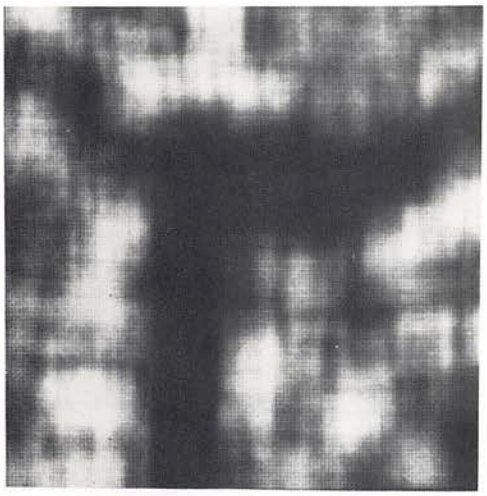
Fig. 3. Texture boundaries detected on 10*th* plane.



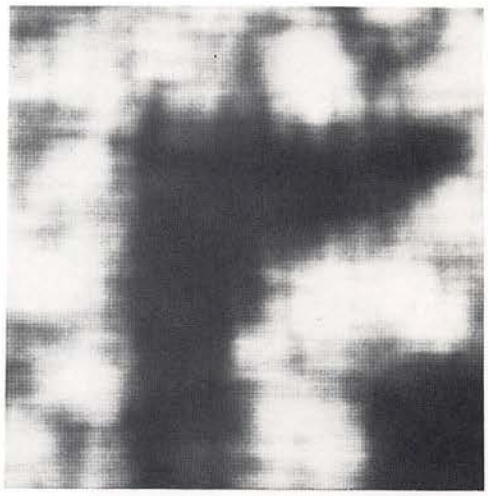
1*st* plane



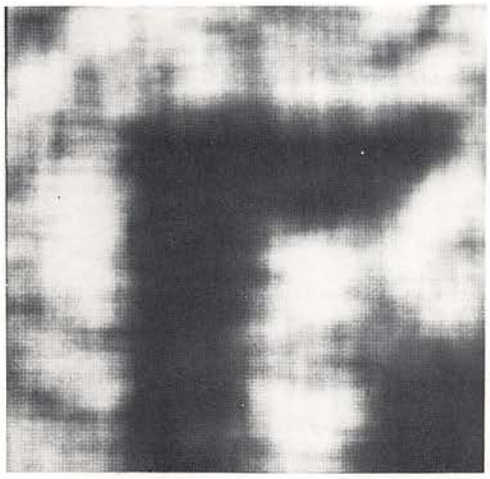
2*nd* plane



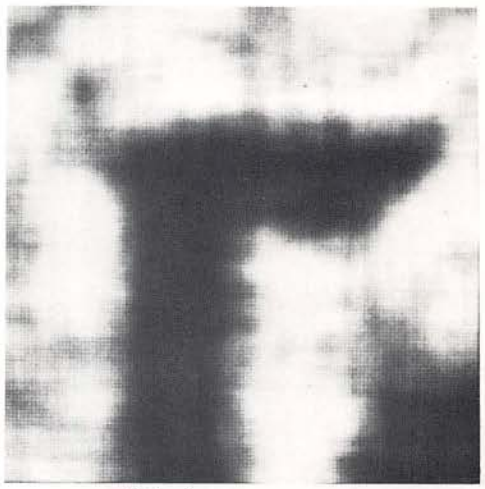
3*rd* plane



4*th* plane

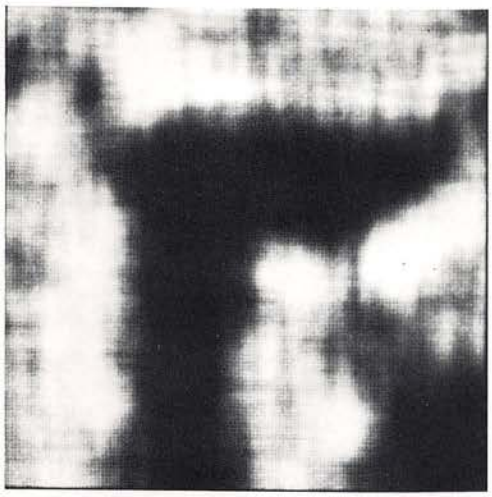


5*th* plane

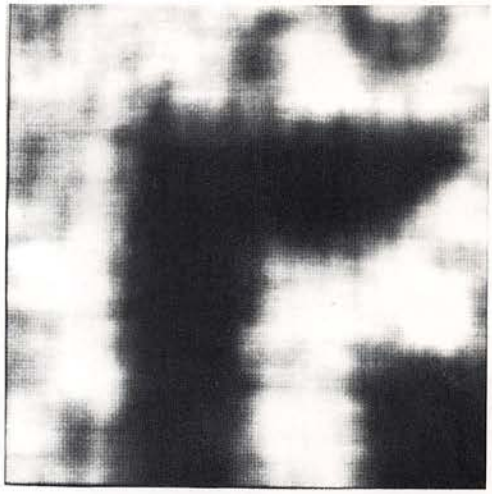


6*th* plane

Fig. 4. 15 available texture energy planes.



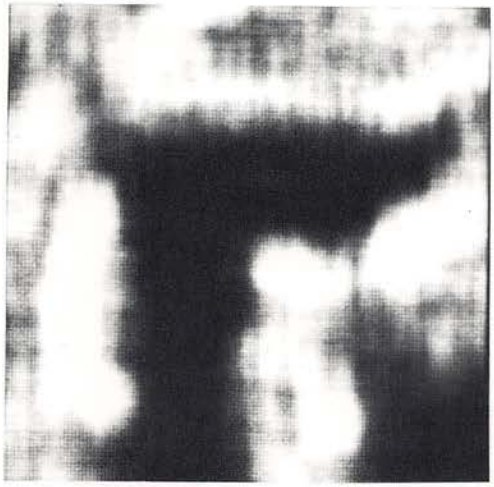
7*th* plane



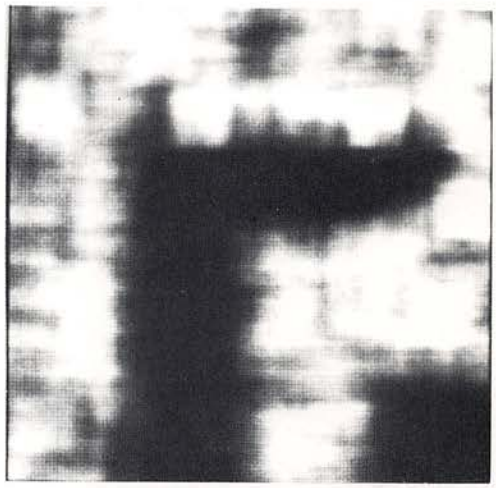
8*th* plane

9*th* plane

10*th* plane



11*th* plane

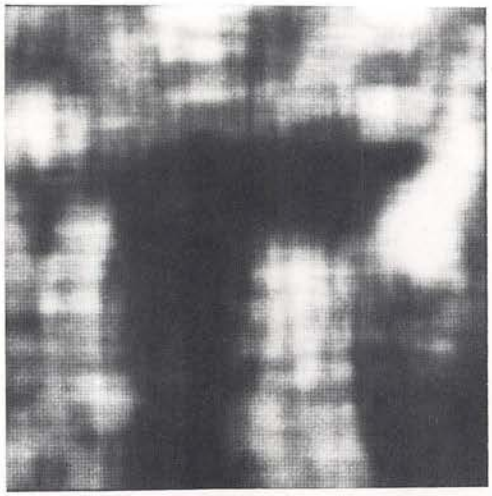


12*th* plane

Fig. 4. (Continued)

13*th* plane

14*th* plane



15*th* plane

Fig. 4 (Continued)

Fig. 5. Hybrid plane

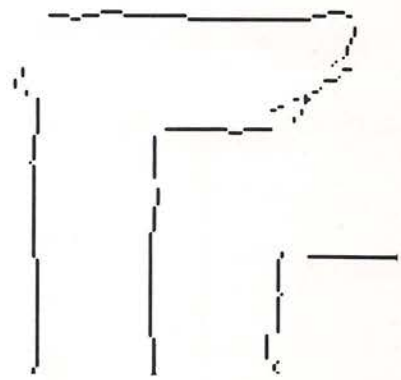


Fig. 6. Texture boundaries detected on hybrid planes.

Table 1. LABELS OF SEGMENTS OF OBJECT (FIG. 3(b)) AT VARIOUS ITERATIONS OF THE MAXIMIZATION OF CRITERIA J^1 AND J^2 AT THE FIRST AND SECOND STAGES OF RELAXATION. IN THE BRACKET WE HAVE INDICATED THE PROBABILITY OF ASSIGNMENT OF THE MOST LIKELY LABEL.

Segments of the Object	Labels and Probabilities at Different Iterations							
	0	4	7	10	4	7	13	19
1	30 (.09)	27 (.21)	27 (.23)	27 (.25)	27 (.29)	27 (.30)	27 (.39)	27 (.52)
2	28 (.19)	28 (.27)	28 (.28)	28 (.30)	28 (.31)	28 (.33)	28 (.51)	28 (.68)
3	25 (.10)	29 (.13)	29 (.13)	29 (.14)	29 (.16)	29 (.17)	29 (.23)	29 (.43)
4	5 (.09)	5 (.14)	5 (.17)	5 (.19)	5 (.20)	5 (.21)	5 (.24)	5 (.28)
5	30 (.09)	30 (.12)	5 (.13)	5 (.15)	5 (.16)	5 (.18)	5 (.23)	5 (.38)
6	30 (.09)	4 (.14)	4 (.17)	4 (.18)	4 (.20)	4 (.21)	5 (.27)	5 (.39)
7	6 (.14)	6 (.28)	6 (.36)	6 (.38)	6 (.39)	6 (.41)	6 (.46)	6 (.54)
8	19 (.09)	30 (.13)	1 (.13)	1 (.15)	1 (.15)	7 (.17)	7 (.25)	7 (.37)
9	30 (.09)	3 (.16)	3 (.17)	3 (.21)	3 (.22)	3 (.24)	3 (.23)	7 (.33)
10	5 (.11)	3 (.18)	3 (.21)	3 (.24)	3 (.25)	3 (.27)	3 (.32)	3 (.30)
11	4 (.27)	4 (.33)	4 (.37)	4 (.38)	4 (.40)	4 (.41)	4 (.48)	4 (.60)
12	30 (.09)	5 (.18)	5 (.19)	5 (.21)	5 (.23)	5 (.27)	5 (.37)	5 (.51)
13	7 (.11)	7 (.20)	7 (.21)	7 (.24)	7 (.25)	7 (.26)	7 (.30)	7 (.31)
14	30 (.09)	8 (.13)	8 (.16)	8 (.18)	8 (.20)	8 (.25)	8 (.28)	8 (.38)
15	30 (.09)	30 (.12)	30 (.12)	30 (.12)	9 (.15)	9 (.18)	9 (.29)	9 (.34)
16	17 (.09)	17 (.15)	17 (.18)	17 (.22)	17 (.25)	17 (.24)	17 (.16)	13 (.21)
17	30 (.09)	30 (.14)	30 (.14)	30 (.16)	30 (.18)	30 (.18)	14 (.21)	14 (.42)
18	30 (.09)	30 (.14)	30 (.15)	30 (.15)	20 (.18)	20 (.19)	20 (.23)	15 (.50)
19	30 (.09)	30 (.14)	30 (.14)	30 (.16)	30 (.18)	30 (.19)	30 (.20)	30 (.36)
20	18 (.09)	29 (.13)	29 (.14)	29 (.15)	30 (.16)	30 (.18)	21 (.23)	21 (.29)
21	1 (.14)	21 (.14)	21 (.15)	21 (.19)	21 (.22)	21 (.25)	21 (.41)	21 (.51)
22	22 (.13)	22 (.15)	22 (.18)	22 (.19)	22 (.22)	22 (.25)	22 (.32)	22 (.40)
23	30 (.09)	25 (.12)	25 (.13)	25 (.14)	25 (.15)	25 (.18)	25 (.21)	25 (.26)
Value of Criteria	-	1.28	1.35	1.50	1.30	1.36	1.53	1.56
		$J^{(1)}$			$J^{(2)}$			

Shape Matching in 3-D

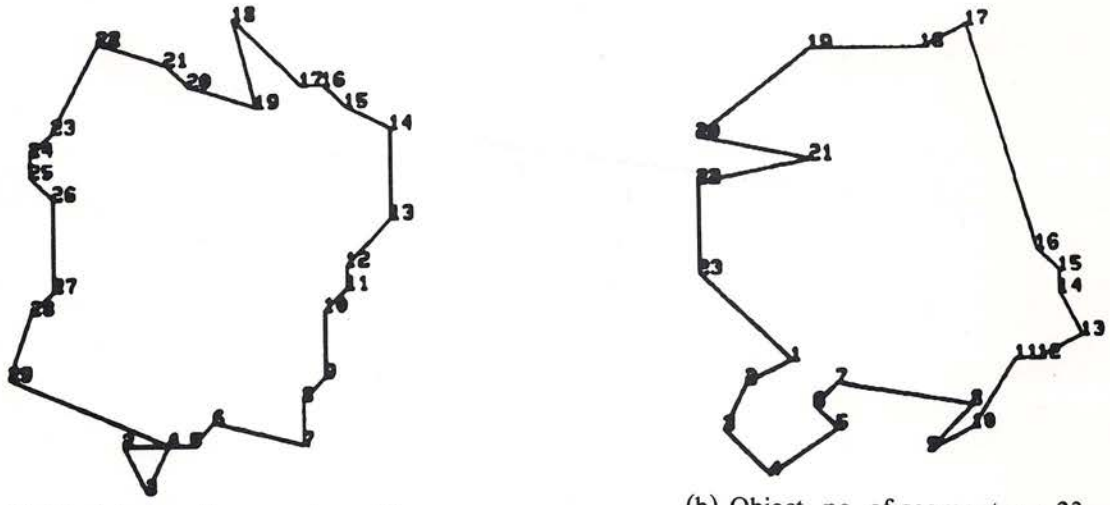
Three dimensional scene analysis consists of providing a model for 3-D objects and a method for matching unknown objects with the model. Given the surface points (x,y,z) of a 3-D object which are not ordered, then first we approximate the surface by polygon faces so as to derive the model of the object. In the sequel we describe an algorithm and present the results on a fairly complex industrial object shown in Fig. 4. This 3-D object was selected because it was easily available and complex enough to simulate the natural environment.

Surface Approximation by Polygon Faces

The points representing the complete surface of the 3-D object are obtained by a sequence of range data images using a laser range finding system (Fig. 5 shows the 14 views of the object shown in Fig. 4) and applying the necessary transformations. The planar convex faces are obtained by choosing three very close points using the three point seed algorithm described below. Details of the algorithm can be found in [6].

1. From the list of surface points select 3 points not on a line and close together in relation to sampling distances.
2. Obtain the equation of the plane passing through the three points picked up in step 1.
3. Find all the points which are very close to this plane.
4. Apply the convexity condition to this set of points to ensure that the set of points make an object face.
5. Check the face obtained in step 4 for narrowness.
6. If the face is obtained correctly (i.e., convexity and narrowness conditions are satisfied), remove the set of points belonging to this face from the list of points and proceed to step 1 with the reduced number of points in the list.

Figure 6 shows the 22 faces obtained for 0^{o} view of the object. Points that could not make up a face having at least 20 points were rejected. Rejected points and the points which are common to two or more faces are shown in different gray levels. In this algorithm first we find large faces and then small faces. An individual view has about 2000 points and was found to have about 10-25 faces. (Horizontal and vertical sampling distances were about 3 and 2 mm, respectively) The complete object was found to have 8314 points after applying the necessary transformations and has 85 faces. We approximate the faces by polygons by following their boundary using a boundary follower and finding the points of maximum curvature on the boundary. The description of a face is given in terms of the properties of the face and its relation to other faces. Neighboring faces are obtained by finding points which belong to more than one face. So the model of the object consists of the description of the faces approximated by polygons and the information about the neighbors of a face.



(a) Model, no. of segments = 29

(b) Object, no. of segments = 23

Fig. 3. Polygonal approximation of the regions shown in Fig. 2 after reducing them by 14 times. A smoothing factor of 4 is used to obtain the polygonal approximations. Figures 3(a) and 3(b) correspond to Figs. 2(a) and 2(b), respectively.

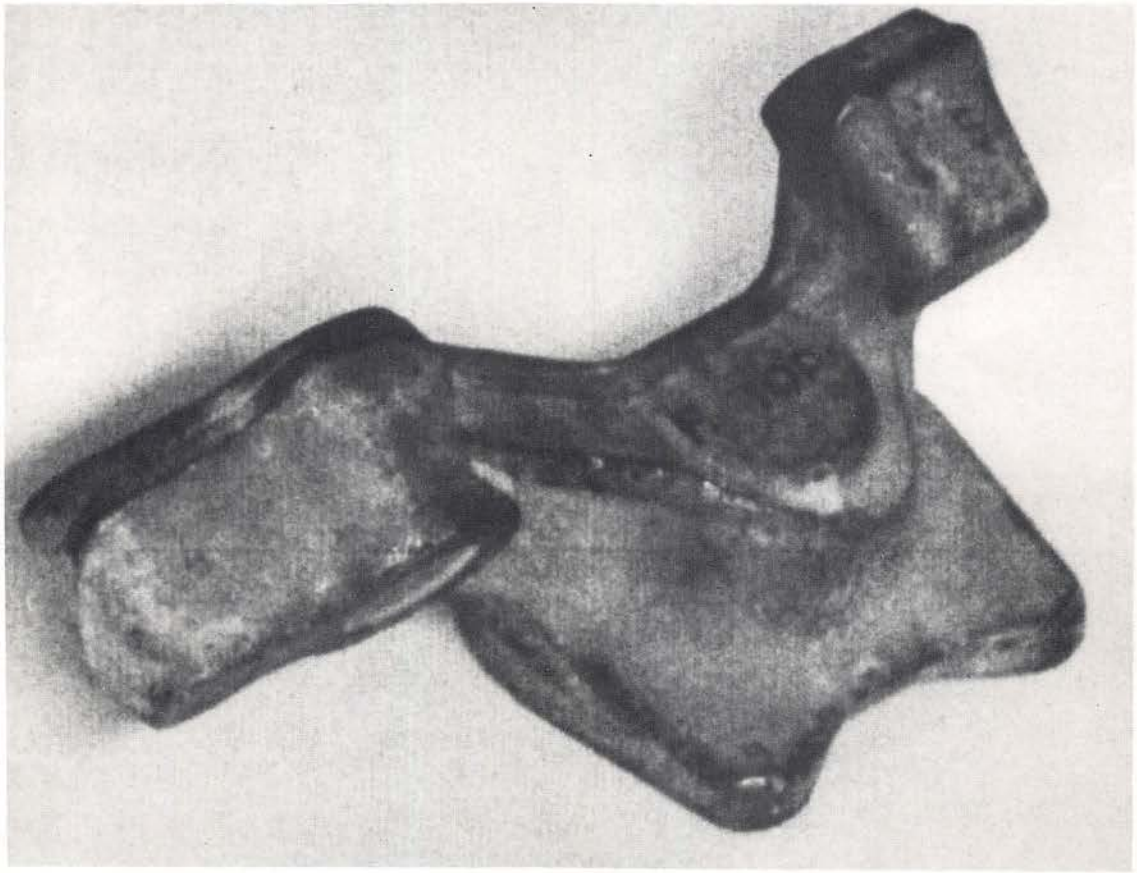


Fig. 4. Automobile piece analyzed.

Fig. 5. The 14 range data views of the object.

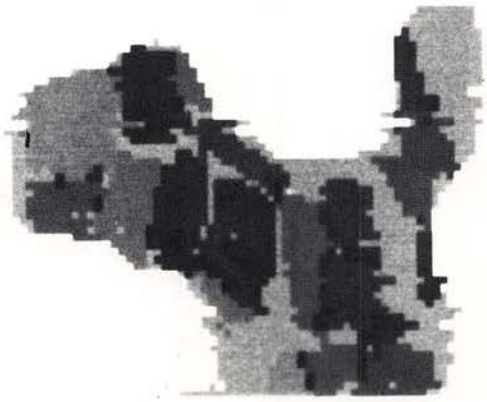


Fig. 6. Faces found in 0° view. Various faces are shown in different gray levels. There are 22 faces in this view. Rejected points and points common to two or more faces are also shown in different grey levels.

Shape Matching of 3-D Objects

Shape matching of 3-D objects is based on matching the faces of the unknown view of the object with the faces of the model using a probabilistic relaxation scheme which explicitly maximizes a criterion functions based on the ambiguity and inconsistency of classification. This criterion is maximized using a projection gradient method and matching is done in a hierarchical fashion [1,5]. At the first level of hierarchy binary relations are used and at the second level a subset of ternary relations are used to compute the compatibility of a face of the unknown view with a face of the model. The first stage does not resolve all the ambiguous labelings. The second stage helps in correcting these labelings. We have found that two levels of hierarchy are sufficient for matching purposes, although the method generalizes to include higher levels at the expense of increased computation.

The description of a face includes:

- Number of points in the face,
- Points on the boundary of the face,
- Number of vertices in the polygonal approximation of the face,
- (x,y,z) centroid of the face,
- Maximum, minimum and average radius vectors from the centroid and,
- Derived features from the above description.

The initial assignment of probabilities of the faces of the unknown view with the model is obtained by using the above features. The compatibility of a face of the unknown view with a face in the model is obtained by finding a transformation and computing the error in the feature values which could be selected so as to make it insensitive to the reasonable changes in the shape. The transformation is based on 1) scale, ratio of area of two faces, 2) translation, difference in the centroidal coordinates of the two faces, 3) difference in the orientation of two faces so that they are in the same plane, 4) rotation, to obtain the maximum area of intercept, once the two faces are in the same plane.

The initial probabilities and compatibilities so obtained are put in the relaxation algorithm which makes use of various strategies that lead to faster computation.

Most of the computation time is taken in finding the faces. Relaxation process takes few minutes. About 6-10 iterations at the first stage followed by an equal number of iterations at the second stage produced reasonable results. Computation time can be cut by the parallel implementation of the algorithm. The matching results could be fed to a robot manipulator on an assembly line or inspection stages of the production.

References

[1] B. Bhanu and O.D. Faugeras, "Shape Matching Using Hierarchical Gradient Relaxation Technique," USCIPI Report 990, pp. 83-113, October 1980.

[2] B. Bhanu and O.D. Faugeras, "Shape Description of Occluded Objects Using Coordinated Hierarchical Gradient Relaxation Method," USCIPI Report 990, pp. 113-137, October 1980.

[3] B. Bhanu and O.D. Faugeras, "Shape Recognition of 2-D Objects," to be presented at *2nd Scandinavian Conf. on Image Analysis*, June 15-17, 1981, Helsinki, Finland.

[4] R. Ohlander, K.E. Price and D.R. Reddy, "Picture Segmentation Using a Recursive Region Splitting Method," *Computer Graphics and Image Processing 8*, 313-333, 1978.

[5] B. Bhanu and O.D. Faugeras, "Shape Matching of 3-D Objects," submitted to *Seventh International Joint Conference on Artificial Intelligence*, Vancouver, Canada.

[6] T.C. Henderson and B. Bhanu, "Three Point Seed Method for the Extraction of Planar Faces from Range Data," submitted to *Seventh International Joint Conference on Artificial Intelligence*, Vancouver, Canada.

2. HARDWARE IMPLEMENTATION OF IU ALGORITHMS

2.1. A Residue Based Image Processor for VLSI Implementation

S.D. Fouse, G.R. Nudd, G.M. Thorne-Booth[2] **and** *P.A. Nygaard* **and** *F.D. Gichard*[3]

ABSTRACT

This paper describes recent work undertaken at Hughes Research Laboratories (Malibu, California) in support of the DARPA Image Understanding (IU) program. The principal goal of the work is to investigate the application of VLSI technologies to IU systems and identify processor candidates well suited to VLSI implementation. One candidate that is very well suited to the VLSI technology is a programmable local area processor with residue arithmetic based computations. The design and development of this processor, which operates on a 5x5 kernel are described. Of significant interest is an LSI custom circuit that we are developing and which will perform the bulk of the residue computations. In addition, an interface that will allow this processor to be controlled by a general purpose host computer (e.g PDP 11/34) is described.

Introduction

Our previous work [1,2] in developing image understanding architectures has concentrated on the analysis of the processing functions required for special purpose LSI primitives. We have developed about 16 fixed and programmable primitives for real-time operation.

The work described here represents a significant shift in emphasis and an increase in capability. Firstly, we have undertaken a detailed design and analysis of a number of complex processing operations including line-finding [3] and texture analysis [4]. This work has been carried out specifically with LSI and VLSI implementation in mind. Hence issues such as chip and function partitioning, data flow, local storage and wordlength are specifically emphasized. The results of the systems analysis and design for these operations are included in section 2. From this work we have been able to configure a fully

---

[2]Hughes Research Labs, Malibu, Ca.
[3]Hughes Carlsbad Research Center, Carlsbad, Ca.

integrated real-time processor for each.

Of equal importance and perhaps greater impact to military systems and robotics, we have configured, designed, and started to fabricate a VLSI processor that can form the basis of a fully programmable image understanding system compatible with commercially available host machines. The architecture itself uses residue arithmetic [5] to provide a highly regular and extendable structure. These issues are of great importance in the emerging VLSI era where design time and the ability to amortize the fabrication cost of many processors are essential elements. The VLSI processor now under development is configured on a single board with multiple copies of a single custom built nMOS chip. Our estimates indicate that the processor will perform between 50% and 75% of the operations for line finding and texture classification. The modular nature of the machine can provide essentially variable precision as discussed in section 3. The single custom-built chip has a complexity equivalent to approximately 6500 transistors. However, with decrease in design rules from the present 5 μm to submicron we can anticipate building a single chip with some 80,000 transistors and design the whole system around four identical chips.

A significant advantage of our approach is the compatibility with general purpose host machines, such as the DEC series, which are widely used in image analysis and understanding. We have therefore spent considerable effort in developing a UNIBUS interface so that the machine can be accessed through the host software. With the addition of the local area logic processor, to be developed in the next phase, we expect to demonstrate a fully programmable real-time processor.

Systems Analysis and Design

The effective exploitation of VLSI technology in image understanding systems requires that the processors developed be used in as wide a range of systems as possible. This requires that a wide variety of systems be analyzed for the purpose of determining commonality. Our approach has been to select three representative systems to analyze: a line finder, a texture analyzer, and a segmentor [6]. Each system was studied and then a directed graph depicting the data flow was produced (7). The directed graphs had nodes that were functionally complex, so the next step was to perform a logic design for the systems to determine the complexity of the nodes and of the systems. The logic design was done for the line-finder and the texture analyzer systems. For details of each design, see USCIPI Report #990 [8]. A brief summary of the results for each system are presented below.

The line-finder system decomposes into four major functions: edge detection, edge thinning, edge linking, and edge tracing.

A design was generated for each of these functions and Table 1 presents the number of gates each

required. Similarly, the texture analysis system decomposed into five major functions: small window convolution (5x5), small window statistical calculation, scaling, large window statistical calculation, linear transformation, Table 2 presents the gate count for each system.

From the results of the directed graph analysis and the logic design, it is obvious that the function that is common to all three systems and the most complex, when measured by the number of gates, is the small window (5x5) convolution. This supports our decision to build a programmable 5x5 local-area processor as the basic VLSI module.

A Residue Based Image Processor

The work described in Section 2 motivated the design of a processor which could perform the computationally intensive low-level operations for each of the three systems investigated. In addition to fulfilling the requirements of the three systems, we also wanted to select an architecture which could be extended to take advantage of the VLSI design and processing capabilities which are currently being developed. The architecture we selected is based on the technique of residue arithmetic.

Processor Description

We implemented our local area processor in residue arithmetic to take advantage of modularity, and hence ease of design, within the VLSI chip and extendability to handle arbitrary dynamic range and accuracy. The technique relies on the conversion, prior to computation, of all the data to relatively prime bases (we chose 31, 29, 23, and 19) and the subsequent decoding of the processed data back to binary numbers. If this overhead is accepted then the arithmetic itself is reduced both in complexity and in required dynamic range. This enables us to use look up tables, which in our case are programmable RAM, to perform the necessary arithmetic. Regularity, ease of VLSI design, and function density are significant advantages. Thus this approach is ideal for VLSI implementation.

A block diagram of a general residue processor is shown in Figure 1. Some of the advantages (e.g. modularity) and disadvantages (encoding, etc.) of this technique are clearly visible in this representation. The encoding and decoding, when compared to a binary processor, are overhead functions and can be the major disadvantage of a residue processor. However this overhead cost can be reduced if enough computations can be performed while in the residue representation, and hence the encoding and decoding amortized over a large computation base. The clear advantage of this type of processor is its natural parallelism. Each parallel computation channel is independent, requiring no communication with its neighbors until the conversion from the residue representation to a binary representation is performed.

Table 1. GATE COUNT FOR LINE FINDER SYSTEM.

10697–12

EDGE DETECTION	178K
THINNING	190 GATES
EDGE LINKING	500
EDGE TRACING	12 MBIT MEMORY +5K LOGIC GATES

Table 2. GATE COUNT FOR TEXTURE CLASSIFICATION SYSTEM.

10697–13

5 LINE KERNEL GENERATION	15K GATES
5 x 5 CONVOLUTION	27K GATES/ CHANNEL
5 x 5 VARIANCE	10K GATES
NORMALIZATION	1K/CHANNEL
LARGE WINDOW STATISTICAL CALCULATION	8.25K/CHANNEL
TRANSFORM (M INPUT CHANNELS)	2.1K·M/OUTPUT CHANNEL

Table 3. FUNCTIONAL CAPABILITIES OF RADIUS.

10697–14

POINT OPERATIONS
POLYNOMINAL FUNCTIONS
CONTRAST ENHANCEMENT
1 - DIMENSIONAL OPERATIONS
INTEGER COEFFICIENT TRANSFORMS
POLYNOMINAL FUNCTIONS
2 - DIMENSIONAL OPERATIONS
EDGE ENHANCEMENT
STATISTICAL DIFFERENCING
LOW PASS/HIGH PASS FILTERING
SHAPE MOMENT CALCULATIONS
STATISTICAL MOMENT CALCULATIONS
INTEGER COEFFICIENT TRANSFORMS
TEXTURE ANALYSIS

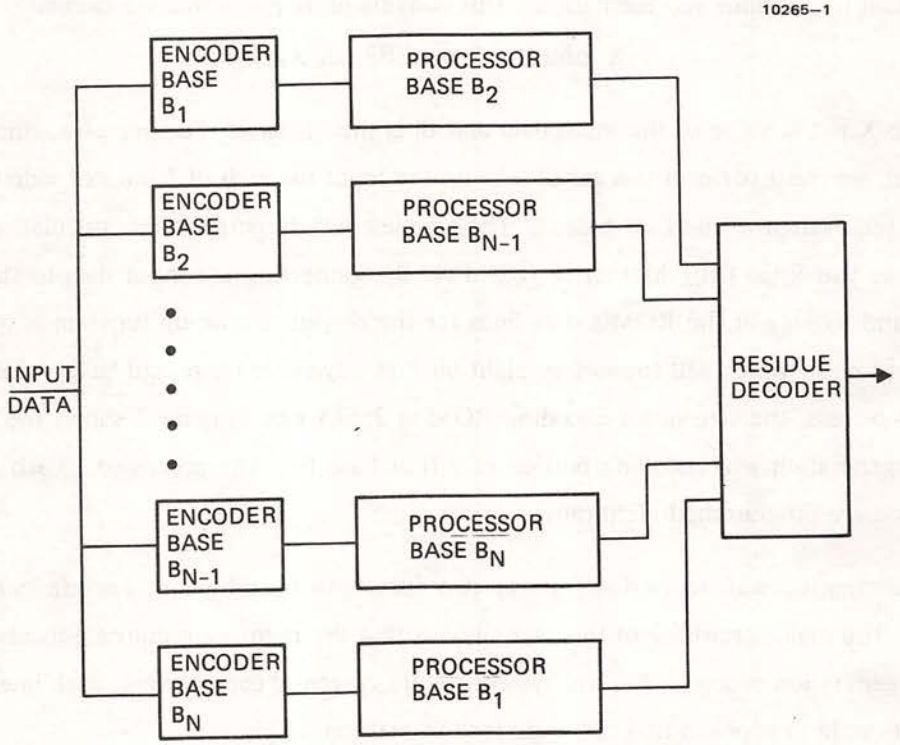


Fig. 1. General structure for Residue Processor

Kernel Generation and Encoding

Typically, the input to an image processor is a string of 8 bit data values generated by a raster scan of the image, in which case we must include in the processor the means for generating the two dimensional kernel. This kernel generation function is most easily accomplished using a series of shift registers. For a five line kernel four shift registers, each one containing as many elements as there are pixels in a line, are required to generate five adjacent lines of video. For our particular application the shift registers are 8 bits wide and 512 elements long.

Before the input data can be processed by a residue processor it must be converted from a binary representation to a residue representation. This conversion requires that we calculate:

$$(X \bmod B1, X \bmod B2, \ldots, X \bmod Bn)$$

where X is the value of the input data and Bi is the ith base. For our case, since we operate on a 5 x 5 kernel, we must perform this calculation on the input for each of 5 lines of video and for each of 4 processors (equivalent to the four bases). The simplest way to perform this calculation for a general set of bases is to use Read Only Memories (ROMs). By connecting the input data to the address lines of the ROM and looking at the ROM's data lines for the output, a look-up function is performed. For our particular processor which will support an eight bit input dynamic range and bases which can be encoded in five bits or less, the size of an encoding ROM is 256x5 bits. Figure 2 shows the block diagram for the kernel generation and encoding portion of a four base five line processor. Each of the five ROMs for each base are programmed identically.

An alternative way to perform these two functions would be to encode before the kernel is generated. The major drawback of this technique is that the memory requirements are much greater for the kernel generation process. For the system we are currently constructing, each line delay would need to be 20 bits wide as opposed to 8 bits wide for the method we chose.

A Programmable Residue Computation LSI Circuit

The actual computations on the image data will be performed by a custom LSI circuit which is currently being processed at the Hughes Carlsbad Research Center. The circuit will process a 5x1 kernel and is capable of performing computations of the form

$$y = f_i(x_i)$$

where y is the output value, x_i are the five elements in the kernel, and f_i represents polynomial functions of a single variable.

A functional block diagram of the circuit is shown in Figure 3. The word size for this is five bits, which limits the prime bases used to a value of 32 or less. The circuit is designed to accept a 5 bit input

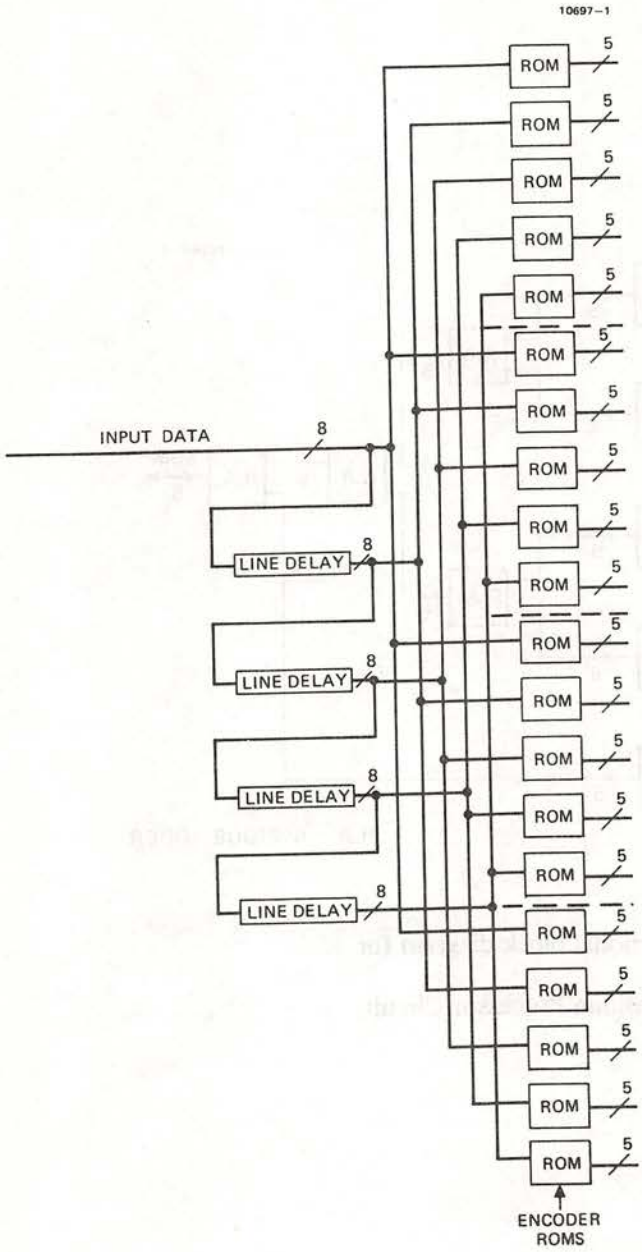


Fig. 2. Kernel generator Encoding for 5x5 Processor.

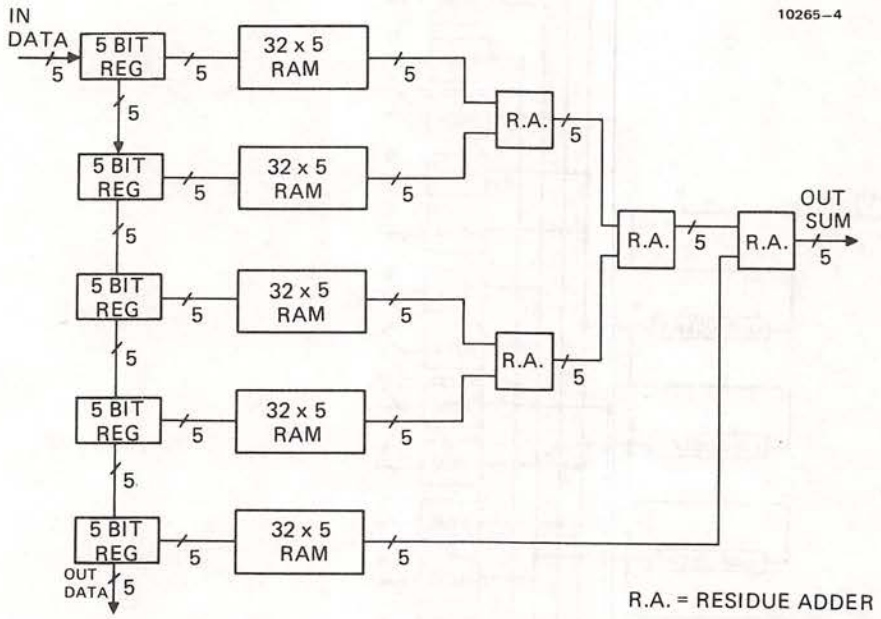


Fig. 3. A functional block diagram for

5x1 Residue Processor Circuit.

word which is clocked into a 5 element shift register. The contents of each register element is then shifted to the next register. The 5 bit data in each of the shift register elements is used to address a look-up table, which is a 32 x 5 Random Access Memory (RAM). This look up operation performs a unary operation, such as a multiplication by a constant or a squaring operation. The outputs of the 5 RAMS are then summed modularly to produce a 5 bit output, the base of the modular addition being programmable by external control of the circuit. Since the look-up tables which perform the unary operation are composed of RAMs the circuit can be programmed for many different computations, such as different weights for a convolution or different powers of a number for a statistical calculation.

A detailed schematic of the circuit is shown in Figure 4. In addition to having 5 bits of input data and 5 bits of output data, an additional set of data lines is included in this design. These data lines, which are bi-directional, serve a multipurpose role for control and testing. When used as input data lines they can be used to program the base of the modular addition and to program any of the 5 look-up tables. When used as output data lines they can read the look-up tables to verify the operation of the circuit.

This circuit is being fabricated using the nMOS technology and has been designed to accept a 10 MHz data rate. To achieve this data rate pipeline techniques were used and the resulting latency for this circuit is 7 clock cycles. The circuit will be packaged in a 28 pin DIP. Figure 5 is a photograph of the layout of th chip, which will be available for testing in April 1981.

To utilize this circuit (with a 5x1 kernel) in a 5x5 local area processor, multiple copies of the circuit need to be used as well as additional logic to combine the outputs of the individual circuits. For each base, 5 of these circuits are used, one for each line of the kernel. In addition four 1024 x 5 bit ROMs are used to sum the outputs of the five circuits. ROMs are used instead of adders because the additions must be done modularly. Figure 6 shows the block diagram of the processor, including the encoding and computation portions.

Decoding

The last portion of the processor is concerned with the conversion from the residue representation to a binary representation. This conversion could certainly be done the same way as the encoding, by table look-up, but there is a severe problem with that approach. For our particular system, to convert four 5 bit values, the decoder would require a memory 1 million elements wide with each element being 17 bits deep. This table is certainly attainable but the approach is not extendable. If an extra base is required, so that five 5 bit values need to be converted, the memory requirements increase to 33 million elements, each being greater than 20 bits deep.

There are two conversion methods that do not require these large memories. One is based on the Chinese

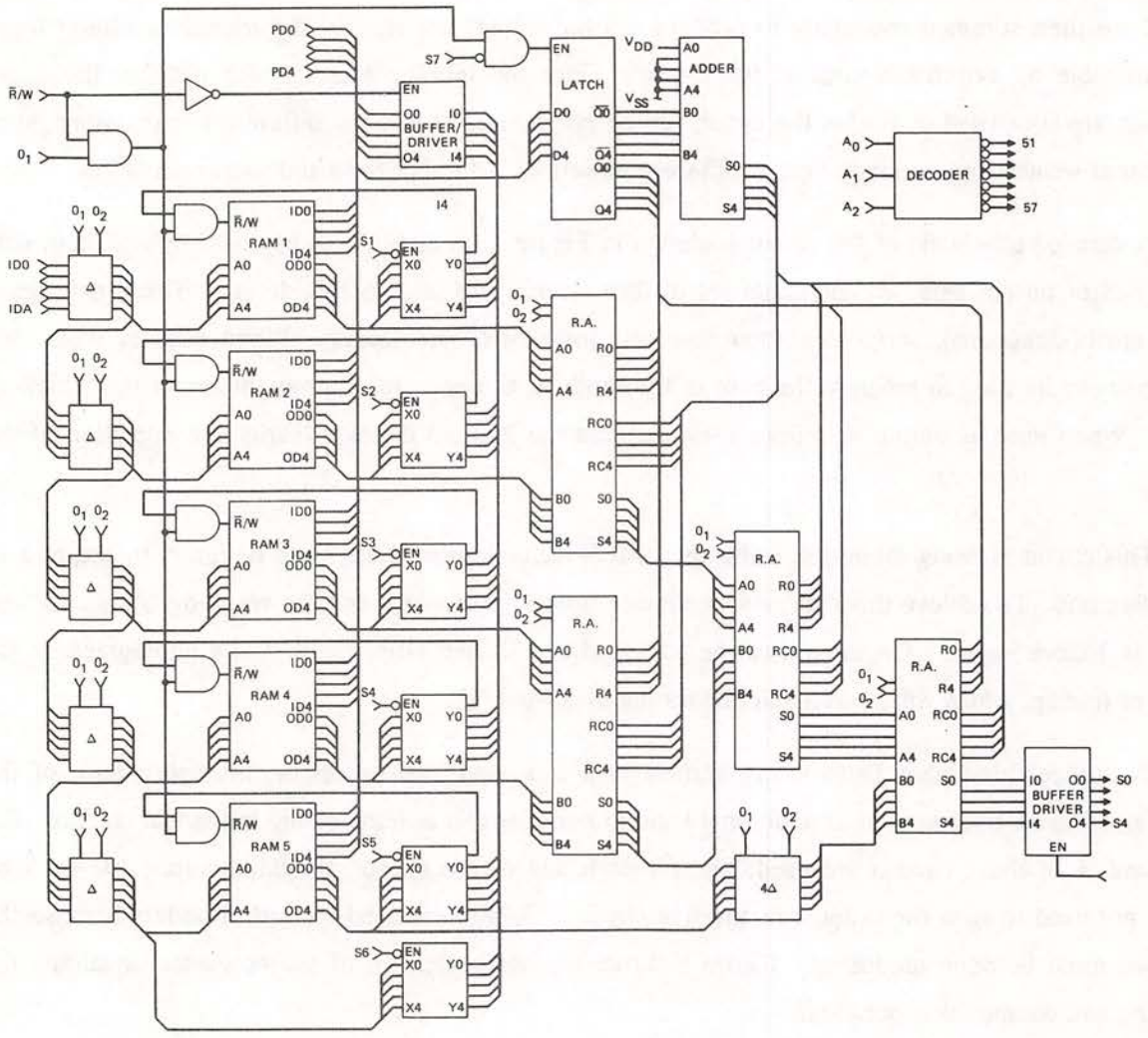


Fig. 4. Schematic of 5x1 Residue Circuit.

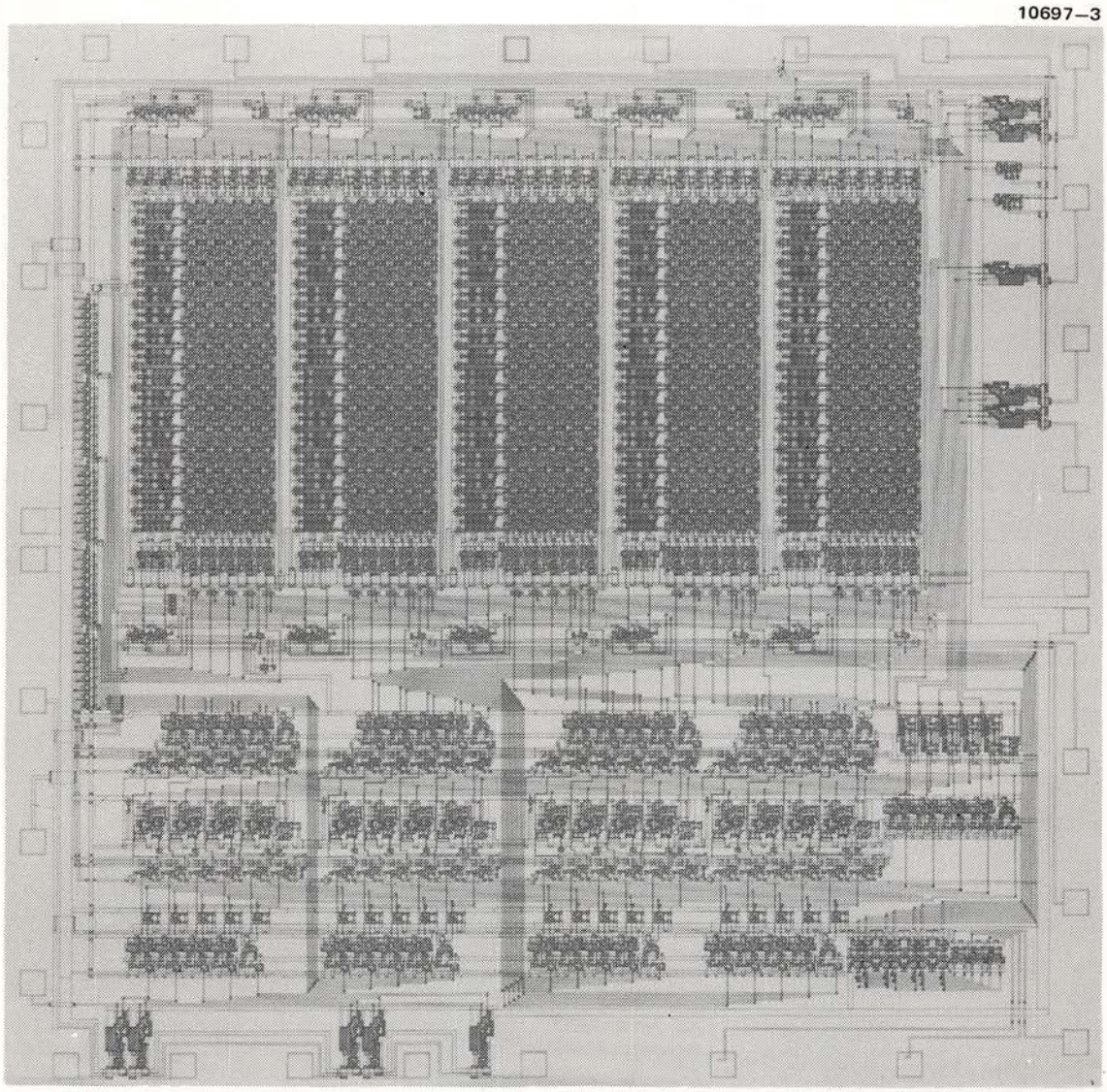


Fig. 5. Photograph of CRC 181 layout.

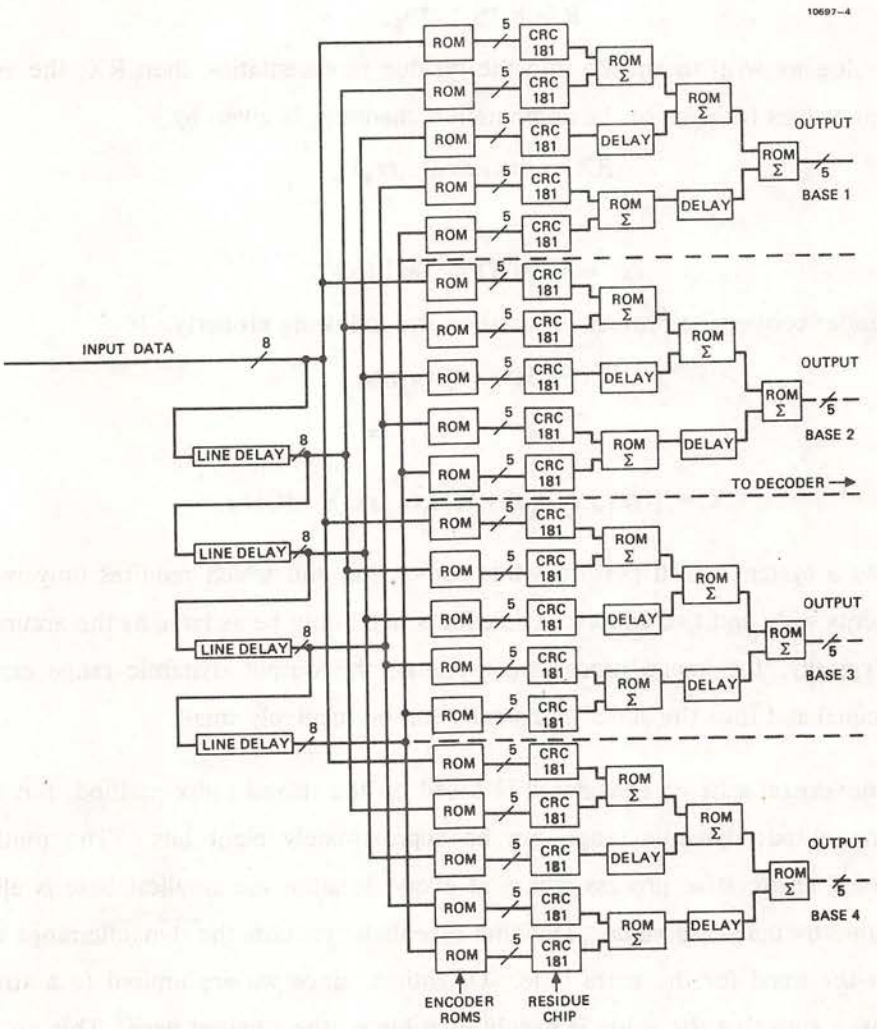


Fig. 6. Structure of 5x5 Processor utilizing 5x1 processor circuits.

Remainder Theorem and the other is based on a mixed radix representation. (For a complete discussion, refer to Ref. [9]) This paper will focus only on the particular implementations of these techniques and the rationale for selecting one over the other.

To be able to reasonably discuss either of the two conversion methods some notation must be introduced. If B is the base vector whose elements are the bases used for the computations,

$$B = (b_1, b_2, \ldots, b_k),$$

R is a scalar whose value is equal to the dynamic range of the processor, which is given by

$$R = b_1 * b_2 * \ldots * b_k,$$

and x is the value we wish to encode into the residue representation then RX, the vector whose elements are the data values for each of the computation channels, is given by

$$RX = (rx_1, rx_2, \ldots, rx_k)$$

where

$$rx_i = x \text{ MOD } b_i, \; i=1 \text{ to } k.$$

The Chinese Remainder conversion process is based on the following property. If

$$RX = (rx_1, rx_2, rx_3, rx_4)$$

then

$$RX = [(rx_1, rx_2, 0, 0) + (0, 0, rx_3, rx_4)] \text{ MOD } r.$$

Figure 7 shows a system which performs this conversion and which requires only two blocks of memory 1024 elements wide and two adders. The adders need only be as large as the accuracy required of the system. Typically, for image processing systems, the output dynamic range and the input dynamic range are equal and thus the adder complexity can be relatively small.

The second conversion scheme considered is based on the mixed radix method, but is simplified by the fact that the output dynamic range can be approximately eight bits. The method can be explained by imagining an iterative process where at every iteration the smallest base is eliminated by dividing the data value by that base value. Dividing essentially reduces the dynamic range of the value and thus eliminates the need for the extra base. Of course, since we are limited to a strictly integer system, we must make sure that the value is evenly divisible by the smallest base. This can be done by rounding up or down so that the element in the residue vector for that base is zero. Figure 8 shows an architecture for a four base system that performs this mixed radix like conversion. At the bottom level of this tree structure the fourth base is eliminated. At the next highest level of the tree the third base is eliminated. Finally we are left with two base values which can be decoded with a simple look-up table. This system has been simulated and the computer programs exist which can generate the contents of the ROMs for this conversion process for an arbitrary set of bases.

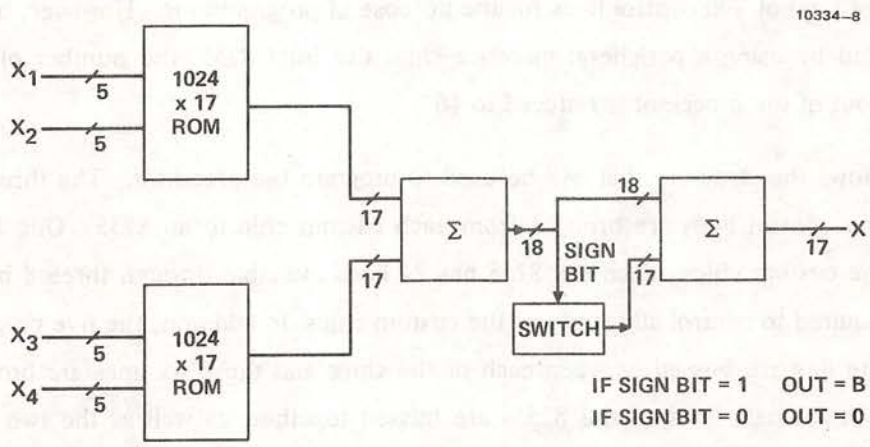


Fig. 7. Chinese Remainder Theorem Residue Decoder (4 base system, 5 bits/base).

We chose the mixed radix based conversion process to be implemented for our processor for two reasons. First, the method does not require any logic other than ROMs. This tends to make it more flexible and reliable. Second, the method appears to be easily extended to more bases, by simply extending the decoding tree. Extending the Chinese Remainder based process would require either larger ROMs or more adders, either way being less attractive way than the mixed radix type conversion.

Programming and Control

A major problem in the fabrication of this processor is gaining access to each of the 20 custom residue chips for the purpose of programming the look-up tables. Each chip has three address lines to select one of five RAM structures, a read/write line, the five bidirectional programming data lines, and a control line for the data line drivers. These 10 control lines must be brought out for each of the 20 custom chips for a total of 200 control lines for the purpose of programming. However, by bussing lines where possible and by using a peripheral interface chip, the Intel 8255, the number of lines that are actually brought out of the processor is reduced to 16.

Figure 9 shows the structure that will be used to program the processor. The three address lines and the bus driver control lines are brought from each custom chip to an 8255. One 8255 is able to control five of the custom chips, since the 8255 has 24 lines available through three 8 bit ports. Thus four 8255s are required to control all twenty of the custom chips. In addition, the five program data lines and the read/write line are bussed between each of the chips and these six lines are brought to a fifth 8255. The eight input data lines of the 8255s are bussed together, as well as the two bit port select address lines. Finally we bring out each of the 5 chip select lines to a binary decoder, allowing selection of a single 8255 using three control lines.

To use this structure to program a given RAM element in the processor requires the following steps. Initially, the fifth 8255 is selected and the data to be programmed are written to the port containing the five program data lines. Next, the 8255 that controls the chip that the desired RAM element is on is selected, and the code to select the desired RAM structure is written to the proper port. Next, the address of the desired RAM element is provided on the input data lines of the processor, and then the address data is shifted so that it is addressing the proper RAM structure. Finally the write data line is strobed to complete the programming sequence. This sequence can be accomplished by three 16 bit data transfers. For a processor using 31, 29, 23, and 19 as the modular bases, a total of 2,550 RAM elements need to be programmed. Thus, if three words transfers are required for each RAM element, a total of 7,650 word transfers are required to completely program the processor.

The processor, which involves all of the functions described above is currently being fabricated. The majority of the electronics will be on a single wirewrap board, but the line delays will be in a

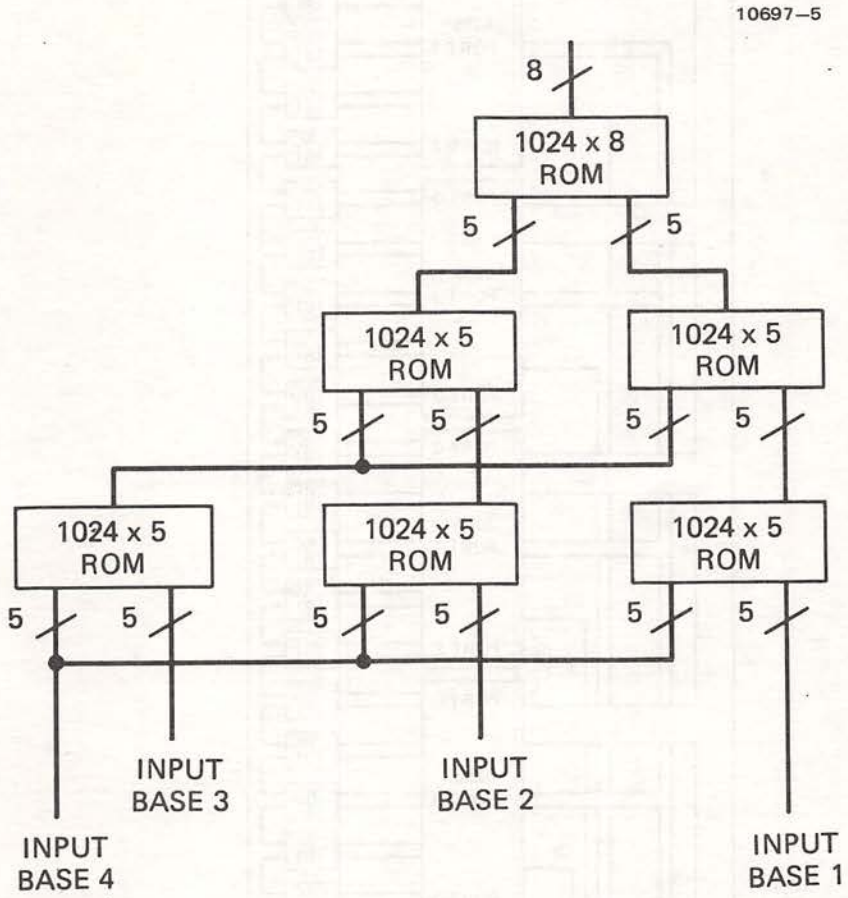


Fig. 8. Mixed Radix Based Residue Decoder
(4 base system, 5 bits/base).

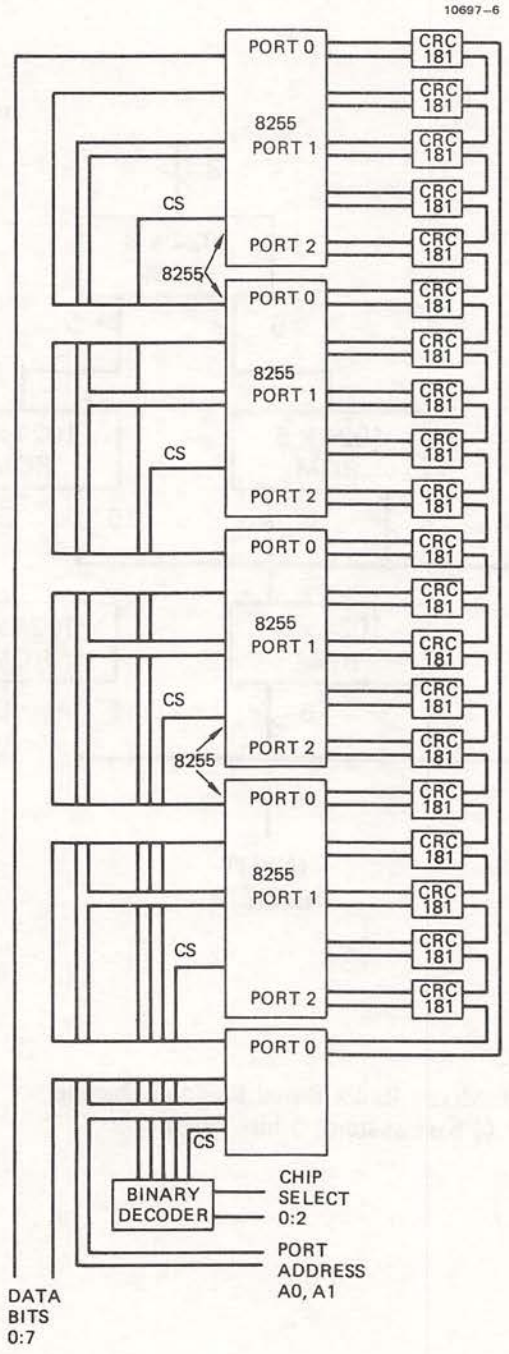


Fig. 9. Structure for programming and Control of Residue Processor.

separate box. A picture illustrating the progress on the wiring of the board is shown in Figure 10.

A VLSI Version of a Residue Computation Chip

Even though the custom chip is performing the bulk of the computations, there is still a fair amount of extra circuitry required. Most of the extra circuitry is necessary because we are using a custom circuit based on a 5 x 1 kernel. The next step then, once this processor is tested and demonstrated, is to develop a 5 x 5 residue custom circuit and fabricate a processor which would utilize these VLSI circuits.

Ideally the 5x5 circuit should include the circuitry for generating the five-line kernel. This would mean that there would be five bits for input and five bits for output. If the kernel generation is performed off of the chip, 25 lines for input would be required or at least a high speed multiplexer would need to be on the chip. On the other hand, if a simple line delay is used to generate the five-line kernel, the circuit would be too inflexible, since it may not be suited to certain applications. This can be avoided by augmenting the shift register with logic to control the clocking. By multiple clocking the shift registers can be made to delay any length up to the maximum length. In other words we could construct an elastic delay line.

Figure 11 shows a block diagram for the data flow of a 5x5 residue circuit. This circuit includes four delay lines, probably 512 elements long, 25 registers for generating the 5x5 window, 25 RAM structures, 24 modular adders, and some delay stages. In addition, programming, control, and testing of this circuit must be considered. A great deal can be learned about these issues by using the processor currently being fabricated.

Both the current chip, the 5x1, and the next chip, The 5x5, have been sized to get a quantitative measure of their complexity. The CRC 181 has a device count of approximately 6,500, of which the RAM portions of the circuit take up 4,500 devices. For the 5 x 5 custom circuit the device count will increase to 80,000. One of the reasons for the high device count is the addition of the line delays, which account for 50,000 devices (for static memory cells). The total number of devices for random logic is about 7000, which is low considering that the circuit will have a throughput of 500 million operations per second.

As stated, going to a 5x5 circuit will greatly reduce the extra circuitry required to construct a 5x5 processor. Figure 12 shows a block diagram of a system utilizing a 5x5 circuit. With the VLSI circuit the package count for the data flow portion of the processor will be only 14. This is compared to a package count in excess of one hundred for the current design. The power and size will be greatly reduced, thereby permitting the processor and the DEC UNIBUS interface to be put on a single card.

MC13901

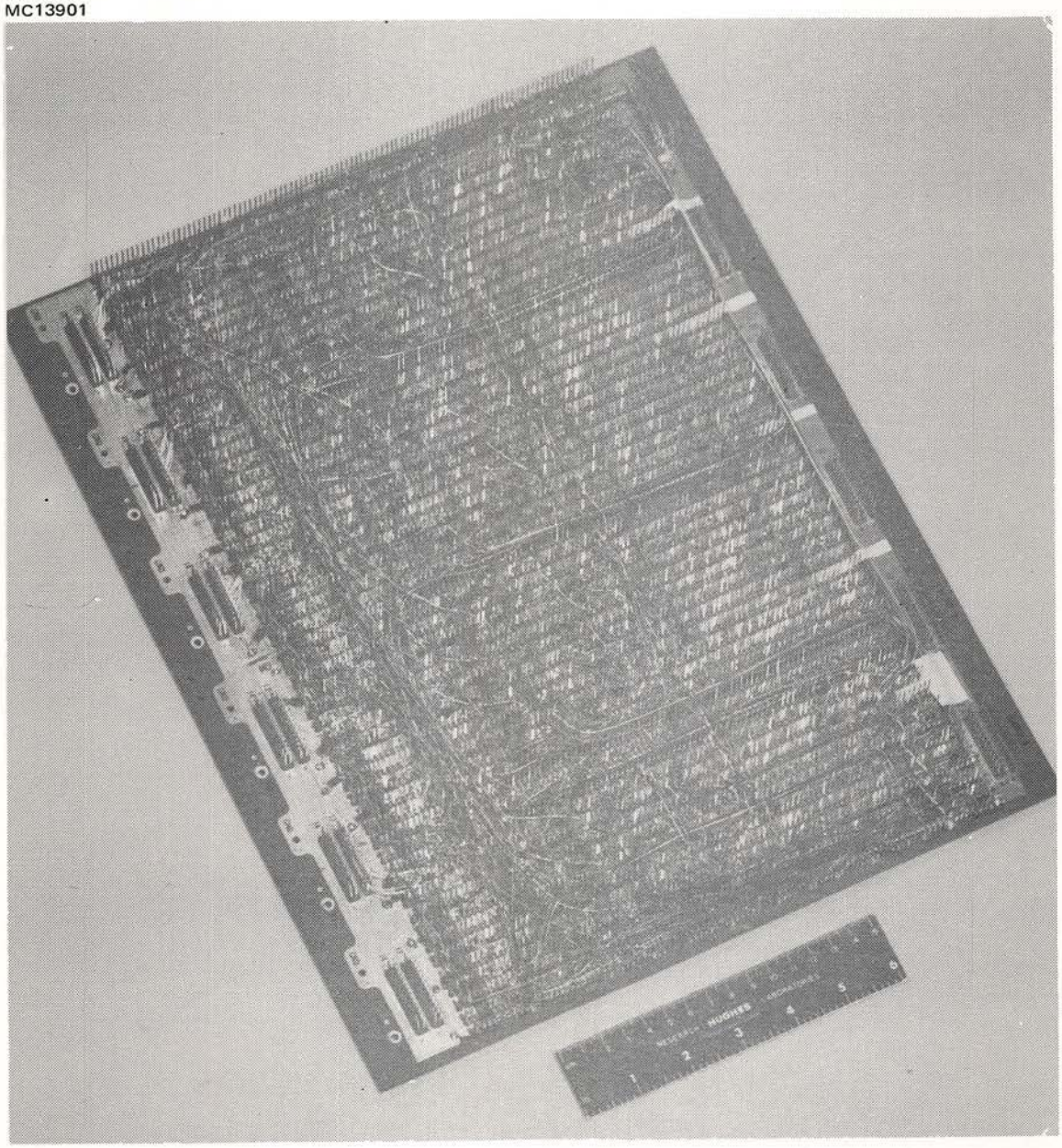


Fig. 10. Photograph of Processor Wirewrap Board.

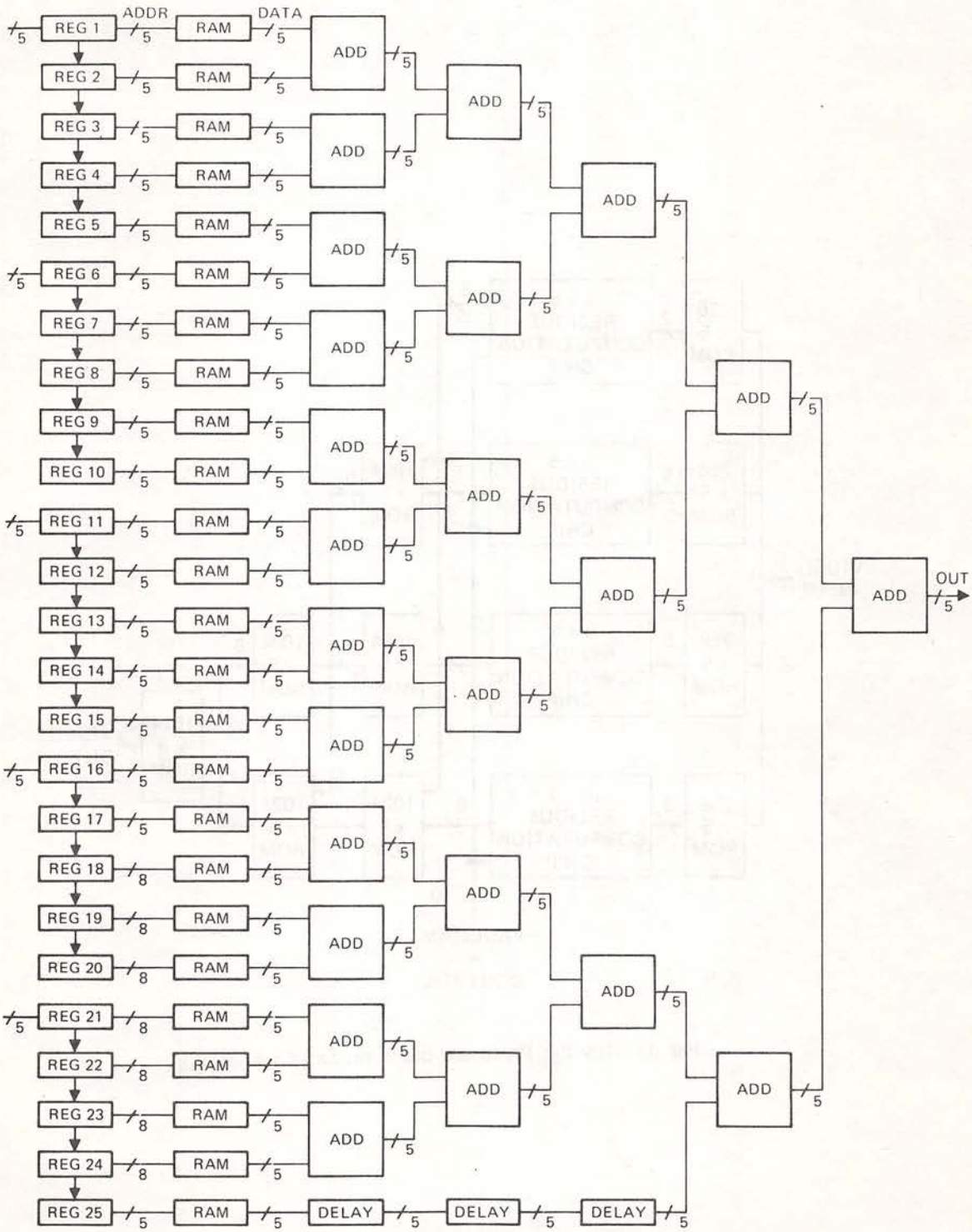


Fig. 11. Data flow for 5x5 Residue Custom Circuit.

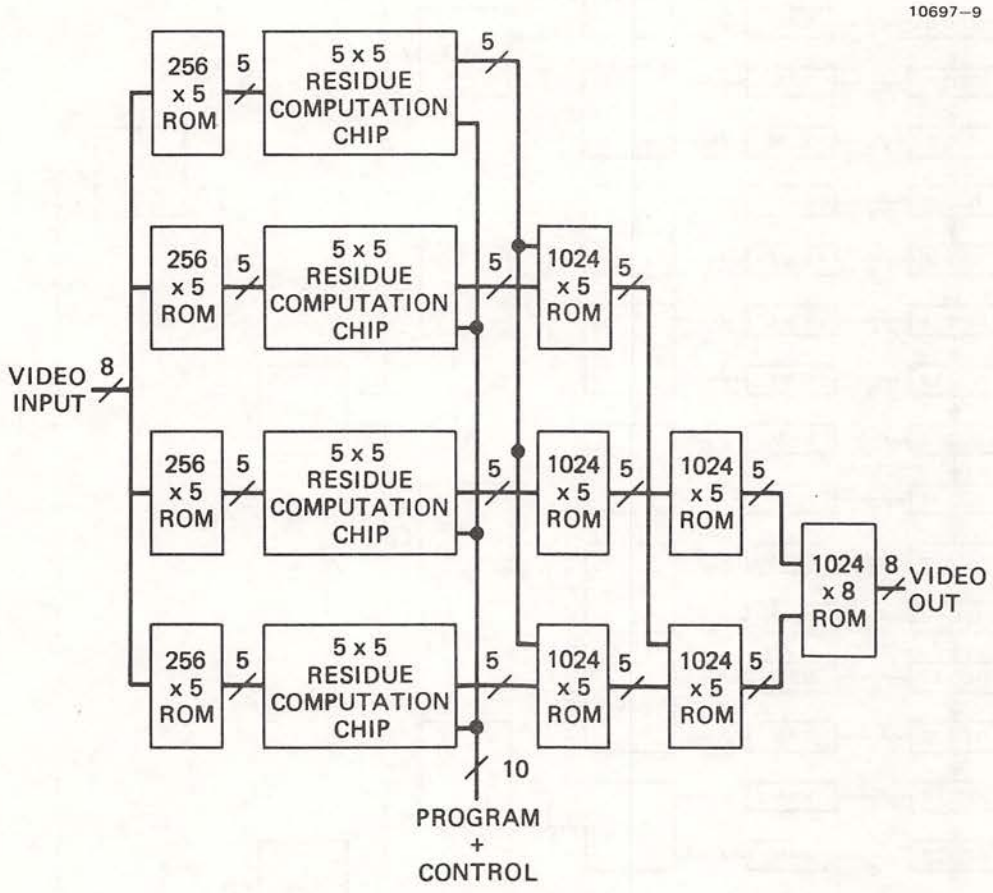


Fig. 12. Residue Processor based on 5x5 Custom chip.

Functional Capabilities

Although the primary motivation for developing this processor was that the systems we investigated required 5x5 convolutions, the processor is capable of performing a wider range of computations. The reason for this flexibility is due to the fact that we used a lookup table to perform a unary operation and the table is completely programmable. The general form of the computation that can be performed by the processor is:

$$y = f_i(x_i)$$

where y is the output, the x_i are the 25 elements in the 5x5 kernel, and f_i are polynomial functions of a single variable. Each f_i is completely arbitrary and need not have any relation to the other f_i. By selecting subsets of the f_i to be identical to zero, we can program the processor to perform point transforms, 1-D transforms of any size up to 5x1, and 2-D transforms of any size up to 5x5. Table 3.1 lists some of the functions that can be performed by this processor.

Unibus Interface

The processor, as mentioned before, is designed to accept data at 100 nanosecond intervals. The reason for this high speed design is to allow real-time stand alone operation. This means, however, that when the processor is used as a peripheral device attached to a general purpose computer, the data transfer will be limited by the memory cycle of the general purpose computer and not by the processor speed. This means, that to get optimal use of the processor, we need the fastest type of transfer available between the processor and the main memory, where the data to be processed will reside. The Direct Memory Access (DMA) type of transfer is the fastest type of transfer that a general purpose computer can support, since it does not require processor intervention. For DEC UNIBUS applications, the fastest data rate one could expect is approximately 1 MHz.

The type of interface we design should then be able to provide a DMA transfer capability for both the programming data and the image data. For either type of transfer, it is essential that the interface be controlled to select the memory location from which the data are to be transferred and to select the number of words to transfer. For program data transfers the interface will only be required to transfer one way at any time. The transfer will be to the processor while in a program mode and from the processor while in a test mode. For image data transfers the interface must be able to transfer data both ways, for input and output. The simplest alternative to handle this bi-directional transfer of data (from a hardware point of view) is to transfer the output data to the same memory location the input data came from, i.e. write the output image over the input image.

DEC devices exist that can provide the DMA transfer capability as well as provide several control

lines to the peripheral device to allow multiple transfer modes. One such device is the DEC DR11B UNIBUS parallel interface. Our plan is to use this device to provide the DMA capability and to design a custom interface to permit the specific transfer modes. The arrangement suggested is shown if Figure 13.

The custom interface will need to interpret the control lines from the DR11B and decide if the transfer is for program data or image data. If it is program data the interface will simply pass the data to the 16 program data lines. If it is an image data transfer, the the custom interface is more complex. Since it is a 16 bit transfer, the data will contain two pixels. So following the transfer, the interface must first pass one byte to the input data lines and then the next byte. Simultaneously the interface must load the first output image data into one byte of the 16 bit output data register and then the next output data into the other byte of that register. Finally the output data registers contents are transferred to the DR11B which writes it to main memory. A preliminary schematic of a system that can perform these types of transfers is shown in Figure 14.

Summary and Future Work

We have described the work undertaken to design VLSI processors for these widely used systems: line-finding, texture classification, and segmentation. From this we believe we can, if required, build the necessary hardware. However, of greater impact, we have identified and started to build a fully software programmable low-level processor for 5x5 operations. The circuitry described relies on a special purpose VLSI chip with 6500 components. Using this, and the interface designed to hook the processor to commercial general purpose machines, most low-level arithmetic operations over a 5x5 kernel can be performed. This work will continue and the full system demonstrated using our in-house PDP 11/34. We then anticipate making this available to interested government researchers in this field.

Our future plans include the investigation and possible development of a single ultra-high density circuit to include the full processor and the development of a compatible logic processor.

References

[1] G. R. Nudd, "Image Understanding Architectures." National Computer Conference, May 1980, Anaheim, CA. *AFIPS Conf. Proc.* Vol 49 pp. 377-390.

[2] S. D. Fouse, G. R. Nudd, and P. A. Nygaard, "Implementation of Image Pre-processing Functions Using CCD LSI Circuits". *Proc. Society Photo-Optical and Instrumentation Engr.* Vol. 225 pp. 118-130, Spie Conf April 1980 Washington, D.C.

[3] R. Nevatia and K. R. Babu, "An Edge Detection, Linking, and Line Finding Program," USCIPI Report No. 840, Sept. 1978.

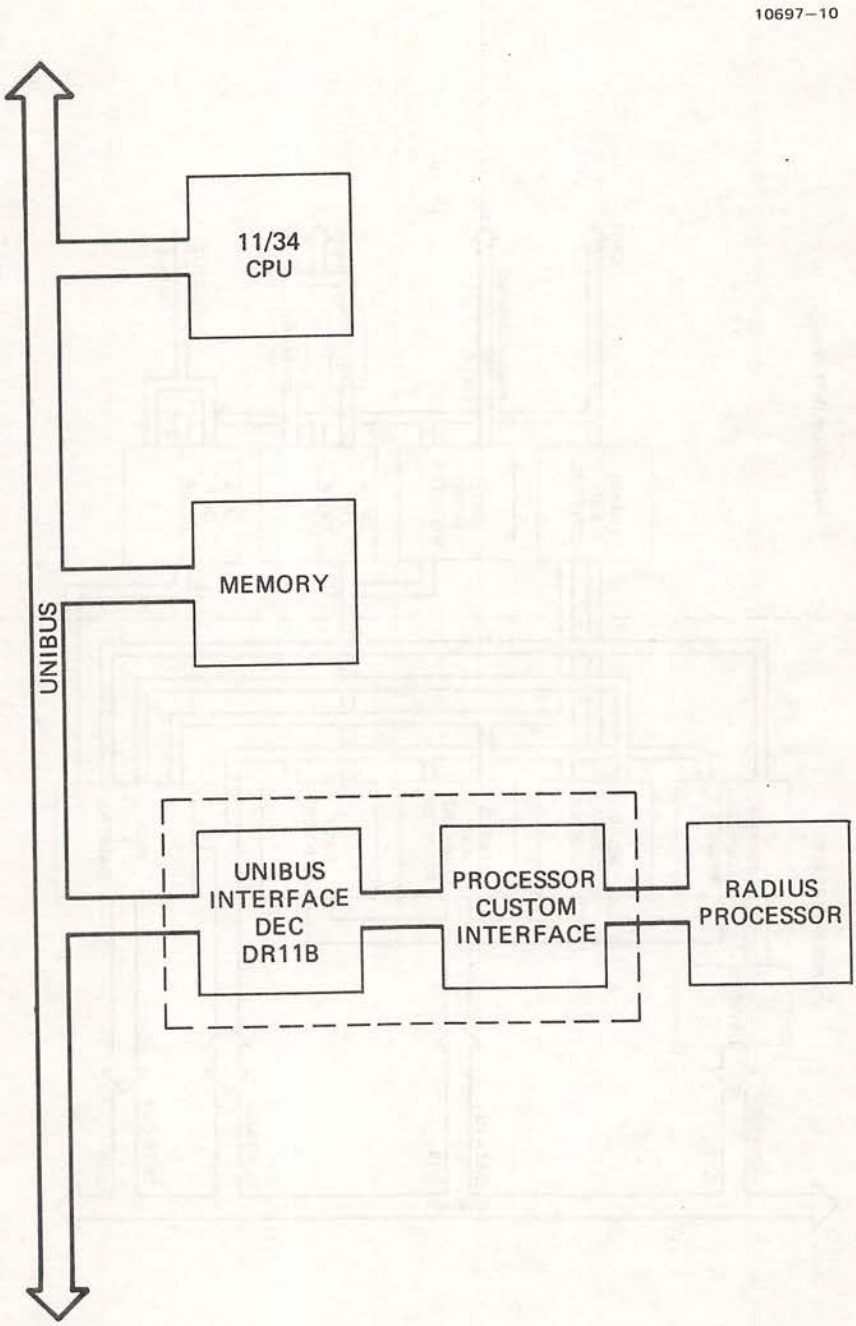


Fig. 13. Commercial/Custom UNIBUS-Processor Interface.

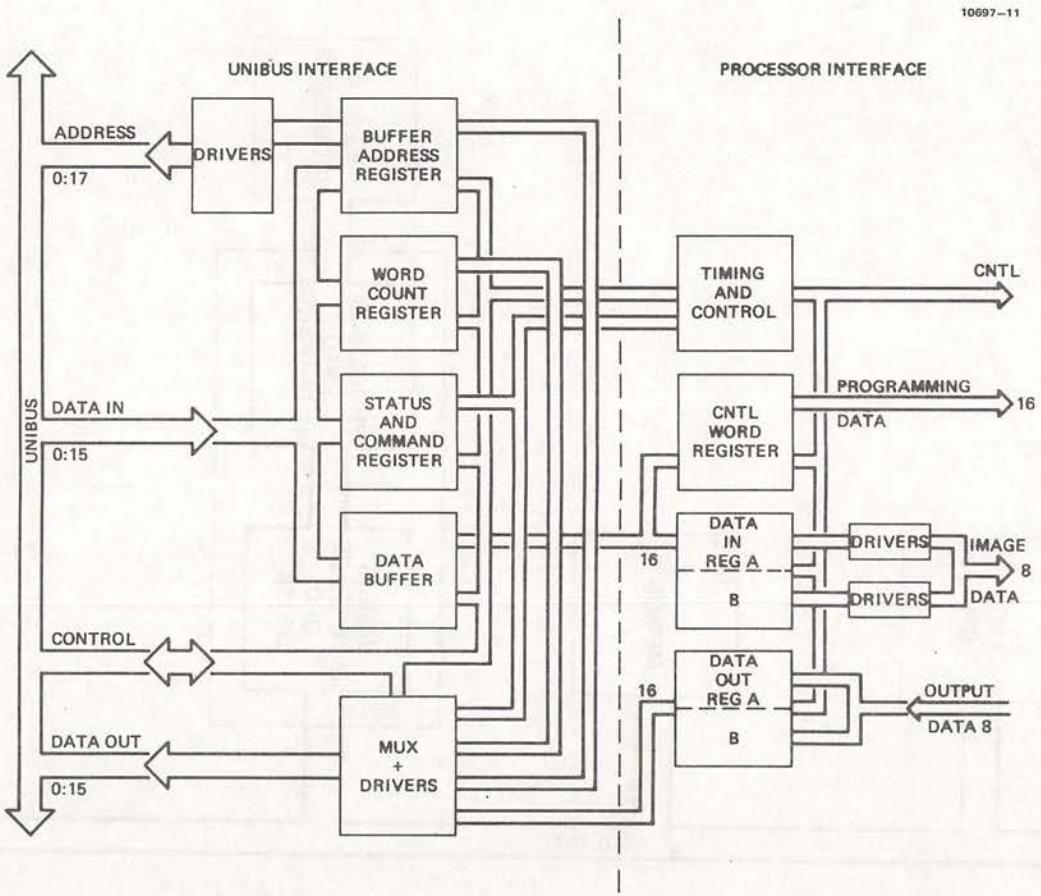


Fig. 14. Bus structure of UNIBUS-Processor Interface.

[4] K. I. Laws, *Textured Image Segmentation*, Ph.D. Thesis, USC, Electrical Engineering Dept., January, 1980

[5] N. Szabo and R. Tanaka, *Residue Arithmetic and its Applications to Computer Technology*, McGraw-Hill, New York, NY, 1967

[6] R. Ohlander, K. Price, D. Raj Reddy, 'Picture Segmentation Using a Recursive Region Splitting Method', *Computer Graphics and Image Processing*, 1978.

[7] S. D. Fouse, G. R. Nudd, V. S. Wong, 'Application of LSI and VLSI to Image Understanding Architectures', *Proceedings Image Understanding Workshop*, April, 1980.

[8] S. D. Fouse, V. S. Wong, and G. R. Nudd, 'Advanced Image Understanding Using LSI and VLSI', USCIPI Report 990, September 1980, pp. 164-204.

[9] A. Huang, *Number Theoretic Processors: A C Array Architecture*, Ph.D. Thesis, Stanford, October 1980.

3. RECENT INSTITUTE PERSONNEL PUBLICATIONS AND PRESENTATIONS

[1] B. Bhanu and O.D. Faugeras, "Description of Occluded 2-D Objects Using a Coordinated Hierarchical Relaxation Technique," to be presented at *2nd Scandinavian Conference on Image Analysis*, June 15-17, 1981, Helsinki, Finland.

[2] B. Bhanu and O.D. Faugeras, "Shape Matching of 3-D Objects," submitted to *Seventh International Joint Conference on Artificial Intelligence*, Aug. 24-28, 1981, Vancouver, B.C., Canada.

[3] B. Bhanu and T. Henderson, "Three Point Seed Method for the Extraction of Planar Faces from Range Data," submitted to *Seventh International Joint Conference on Artificial Intelligence*, Aug. 24-28, 1981.

[4] B. Bhanu and O.D. Faugeras, "Reconnaissance de Formes Planes Par Une Methode Hierarchique D'Etiquetage Probabiliste," to be presented at *AFCET, 3 eme Congres, Reconnaissance Des Formes et Intelligence Artificielle*, Sept. 16-18, 1981, Nancy, France.

[5] O.D. Faugeras and K.E. Price, "Semantic Description of Aerial Images Using Stochastic Labeling," *IEEE-PAMI* Accepted for publication.

[6] O.D. Faugeras and K.E. Price, "Semantic Description of Aerial Images Using Stochastic Labeling," *Proc. ICPR-1980*, Miami, Fl., Dec. 1980, p. 352.

[7] O.D. Faugeras and D.D. Garber, "Algebraic Reconstruction Techniques for Texture Synthesis," *Proc. of Fifth Int. Conf. on Pattern Recognition*, Miami, Fl., Dec. 1-4, 1980.

[8] D.D. Garber and A.A. Sawchuk, "Computational Models for Texture Analysis and Synthesis," *Proc. of SPIE's Technical Symposium East '81, Techniques and Applications of Image Understanding*, Vol. 281, April 1981.

[9] D.D. Garber and A.A. Sawchuk, "Texture Simulation Using a Best-Fit Model," *Proc. of 1981 IEEE Computer Society Conference on Pattern Recognition and Image Processing*, Dallas, Aug. 3-5, 1981.

[10] R. Nevatia and S. Inokuchi, "Boundary Detection in Range Picture," *Intl. Conf. on Patt. Recog.*, Proc. of Miami Beach, Dec. 1980.

[11] R. Nevatia, with C. Clark *et al.*, "Matching of Natural Terrain Scenes," *Intl Conf. on Patt. Recog.*, Proc. of Miami Beach, Dec. 1980.

[12] R. Nevatia, member of Panel on Image Understanding, Int. Conf. on Patt. Recog., Miami Beach, Dec. 1980.

[13] R. Nevatia and A.A. Sawchuk, "Progress in the USC Image Understanding Program," *Proceedings Image Understanding Workshop*, Washington, D.C., April 1981.