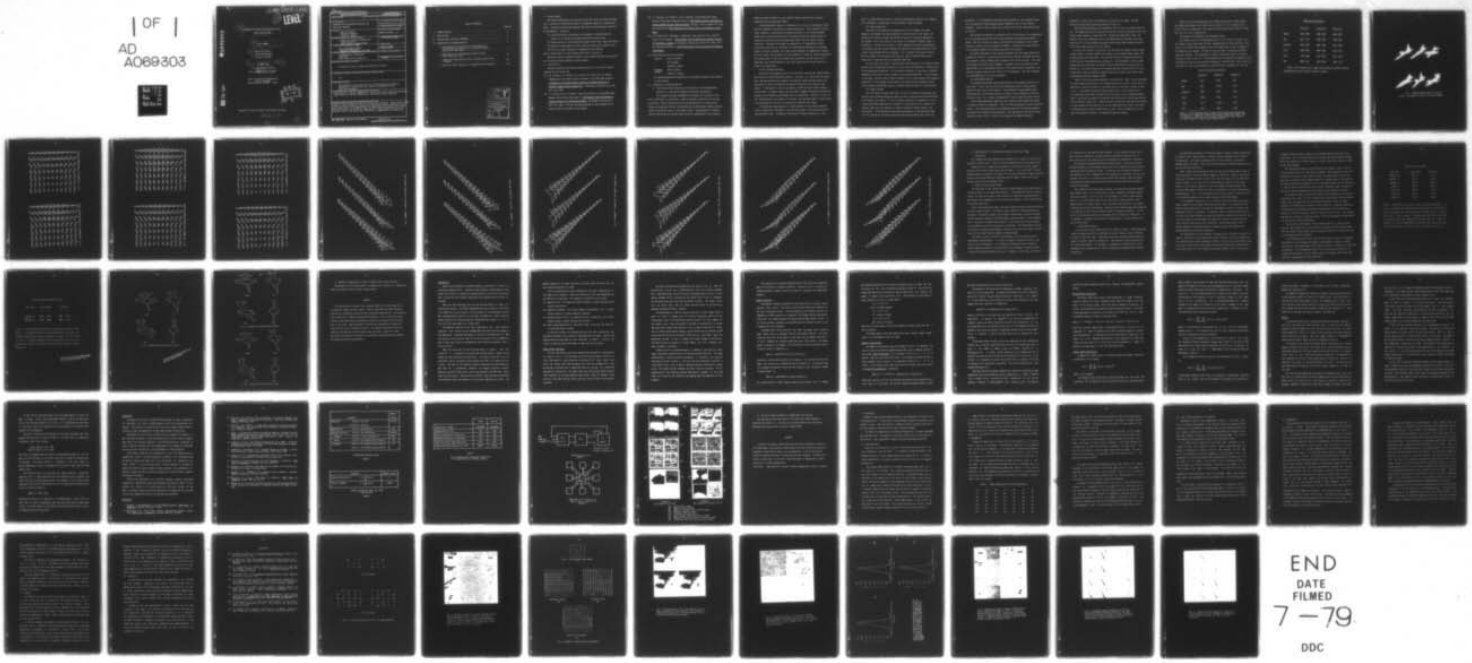
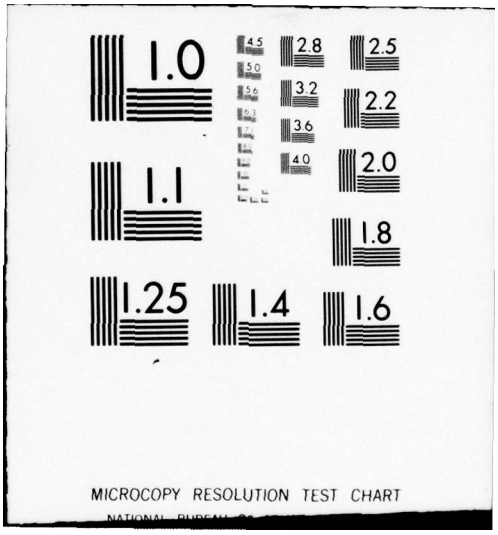
AD-A069 303 PURDUE UNIV LAFAYETTE IND SCHOOL OF ELECTRICAL ENGI--ETC F/G 17/5
SEGMENTATION AND STRUCTURE ANALYSIS FOR REAL-TIME TRACKING. (U)
APR 79 O R MITCHELL, K FUKUNAGA, A P REEVES DAAG29-77-G-0233
UNCLASSIFIED ARO-15541.2-R-EL NL



END
DATE FILMED
7-79
DDC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
MICROCOPY RESOLUTION TEST CHART

(18) ARO (19) 15541.2-R-EL

(12) 13 p.s.

LEVEL II

(6) SEGMENTATION AND STRUCTURE ANALYSIS FOR REAL-TIME TRACKING.

(7) FINAL REPORT. 28 Sep 77 - 31 Jan 79,

(10) Owen R./Mitchell
Keinosuke/Fukunaga
Anthony P./Reeves

(11) 2 April 2, 1979 (12) 66p.

U.S. Army Research Office

(15) Grant Number DAAG29-77-G-0233

School of Electrical Engineering
Purdue University
~~West~~ Lafayette, Indiana 47907

292000

LB

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) SEGMENTATION AND STRUCTURE ANALYSIS FOR REAL-TIME TRACKING		5. TYPE OF REPORT & PERIOD COVERED Final 28 Sep.77 to 31 Jan 79 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Owen R. Mitchell Keinosuke Fukunaga Anthony P. Reeves		8. CONTRACT OR GRANT NUMBER(s) DAAG29-77-G-0233 New
9. PERFORMING ORGANIZATION NAME AND ADDRESS School of Electrical Engineering Purdue University West Lafayette, Indiana 47907		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS U.S. Army Research Office Post Office Box 12211 Research Triangle Park, NC 27709		12. REPORT DATE April 2, 1979 13. NUMBER OF PAGES
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Instrumentation Directorate STEWS-ID-T White Sands Missile Range, NM		15. SECURITY CLASS. (of this report) 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

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

NA

18. SUPPLEMENTARY NOTES

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

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

Shape analysis, tracking, segmentation, Fourier descriptors, parallel processing, image processing, pattern recognition, registration, motion detection.

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

Several results related to real-time tracking are presented. These are: (1) classification of three-dimensional objects and calculation of their orientation using Fourier descriptors; (2) shape analysis of three-dimensional objects using local shape descriptors; (3) sequential image analysis using a parallel binary array processor; and (4) the use of edge information in segmentation and tracking.

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

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

TABLE OF CONTENTS

I. Problem Studied

The problem studied over this contract period was the use of digital processing in locating and identifying objects in sequential imagery for tracking purposes. Four major results have been derived, each of which have been submitted for publication. These are:

(1) Classification and calculation of orientation of three-dimensional objects in a tracking environment using Fourier descriptors;

(2) Shape analysis of three-dimensional objects using local shape descriptors;

(3) Sequential image analysis using a parallel binary array processor; and

(4) The use of edge information in segmentation and tracking.

Each of these topics will be discussed more fully in Section IV.

Additional funds to develop solutions to the tracking problems were supplied by the Defense Advanced Research Projects Agency.

II. Publications

Present and pending publications derived wholly or in part from research sponsored by this grant are:

[1] O.R. Mitchell, A.P. Reeves, S.G. Carlton, R.M. Ward, and J.M. Nelson, "Segmentation and Classification of Targets in FLIR and Video Imagery," Proceedings of the Eighth Annual Symposium on Emerging Patterns in Automatic Imagery Pattern Recognition, Gaithersberg, Maryland, pp. 43-54, April 3-4, 1978.

[2] T.P. Wallace and O.R. Mitchell, "Real-Time Analysis of Three-Dimensional Movement Using Fourier Descriptors," IEEE Computer Society Workshop on Computer Analysis of Time-Varying Imagery, Philadelphia, Pennsylvania, April 5-6, 1979. This has also been submitted to IEEE Transactions on Pattern Analysis and Machine Intelligence.

[3] T.P. Wallace, O.R. Mitchell, and K. Fukunaga, "Three-Dimensional Shape Analysis Using Local Shape Descriptors," IEEE Computer Society Conference on Pattern Recognition and Image Processing, Chicago, Illinois, August 6-8, 1979. Also submitted to IEEE Transactions on Pattern Analysis and Machine Intelligence.

[4] A.P. Reeves and A. Rostampour, "Sequential Image Analysis with a Parallel Binary Array Processor," IEEE Computer Society Workshop on Computer Analysis of Time-Varying Imagery, Philadelphia, Pennslyvania, April 5-6, 1979. This has also been submitted to IEEE Transactions on Pattern Analysis and Machine Intelligence.

III. Participating Scientific Personnel

Faculty -	Owen R. Mitchell
	Ken Fukunaga
	Anthony P. Reeves
Graduate Students -	John M. Nelson
	Timothy P. Wallace

No advanced degrees were completed by the graduate students while working on this project.

IV. Presentation of Research Results

A. Classification and Calculation of Orientation of Three-Dimensional Objects in a Tracking Environment Using Fourier Descriptors.

This research has been documented as reference [1] in Section II and will be available in reprint form. Some of the major conclusions will be presented here. Fourier descriptors of the outside contour of a closed planar figure are produced by Fourier transforming the one dimensional complex function found by tracing the figure in the complex plane. The operations of rotation, scaling, and moving the starting point are easily implemented in the frequency

domain by simple arithmetic on the frequency domain coefficients, producing normalized Fourier descriptors (NFD).

A three-dimensional object is represented by a library of projections, whose NFD's are compared to the NFD of an unknown projection. Since these NFD's have a linear property, an interpolation procedure performed in the frequency domain enables more accurate determination of the angle at which the unknown object is viewed than can be achieved by methods which simply find the nearest library projection. Discussed in the paper are methods for (1) computation of NFD's using the discrete Fourier Transform, (2) normalization procedures to eliminate symmetry and noise problems, (3) classification methods for comparing the unknown with the library, (4) estimation of the orientation in terms of library projections, (5) spatial domain filtering to remove chain code error, and (6) the use of a eigenvector data transformation to limit the library entry for each view of an object to 6 coefficients.

(1) The Tracking Algorithm.

We can use the preceding results in an efficient algorithm for identification and tracking of three-dimensional objects. We are not just identifying the object once, and then following its progress, but actually identifying the object with each frame, as well as estimating its present orientation with respect to the camera. If the object is lost by the camera, so that the object being tracked is a false alarm, or another object, the algorithm is capable of quickly realizing the problem, and re-identifying the object and its orientation. Similarly, if the initial identification of the object is incorrect, and the wrong object is being tracked, the program will realize its mistake and re-identify the target.

This continuous shape analysis is clearly helpful in the tracking problem itself, reducing the likelihood of tracking false alarms or the wrong target for long periods of time. In addition, the record of target orientation vs. time

which is created could be useful in analyzing aerodynamical behavior, for example. This information is unavailable with conventional tracking methods.

(2) Target Acquisition

This part of the algorithm is executed on the first frame of the input sequence, and whenever the program decides it has lost lock on the target. We assume that a library has been constructed for each of the objects expected to be tracked. First, the outline of the target is extracted by whatever means is appropriate. The NFD of the contour is then computed, the linear transformation is performed, and the distances between the unknown (transformed) NFD and all the library (transformed) NFD's are computed. The n (typically 10) nearest library NFD's are taken to be possible classifications, and the diamond-shaped region around each is explored with the estimation algorithm.

At this point, the target is identified as that belonging to the nearest library projection or interpolated projection, and the starting orientation is identified as the corresponding orientation. If the resulting distance is less than a threshold depending on the object, control passes to the tracking algorithm. If the distance is too great, the next frame is again processed by this target acquisition procedure. Optionally, if the distance was too great, the target could be declared a false alarm, and some intermediate range of distances could be cause for re-acquisition.

(3) Target Tracking

Once the target has been identified, the initial orientation estimated, and a suitably small distance obtained we enter the fast tracking algorithm which follows the rotation of the object. A list of closest projections is prepared as in the target acquisition algorithm, but instead of it consisting of the n nearest projections taken from the libraries, it merely consists of either three or five projections. If the estimation algorithm was successful on the preceding frame, the list consists of the three projections defining the sector used in the last

estimation. If the estimation algorithm was unsuccessful on the preceding frame, the list consists of the preceding nearest projection, and its four nearest neighbors. There is hence a wider area of search for the next frame when estimation fails.

The estimation algorithm is performed on the list of three or five projections as in the target acquisition, and the best distance or interpolated distance is found. Once this distance is found, we continue this procedure as long as each distance is below a certain threshold (tracking). We execute the target acquisition procedure whenever the distance goes above the threshold.

The speed improvement in the tracking procedure over the target acquisition procedure is two-fold. A major portion of the time required to execute the tracking procedure is spent computing the distances to the library vectors. The tracking procedure eliminates that altogether. Secondly, while the acquisition procedure looks for an interpolated distance in the region around n projections, the tracking procedure looks at only three or five projections. The time to compute the transformed NFD is of course unchanged.

(4) Experimental Results

The algorithm above was tested experimentally on 18 sequential sets of aircraft data. First, a set of aircraft was synthesized using a simple graphics program written by one of the authors. Three-dimensional approximations were constructed for six different aircraft, a Mirage, a Mig, a Phantom, an F104, an F105, and a B57. Fig. 1 shows a representative library image from each aircraft.

These three-dimensional images were then rotated through appropriate angles to create a library of projections. Figs. 2-7 show one quadrant of each projection library. The libraries consist of two quadrants each, but the missing quadrants are merely mirror images of the ones shown. The resolution of the actual library contours is that of Fig. 1; Figs. 2-7 are shown at one eighth the actual

resolution in order to fit 77 projections in one 512 by 512 image. The NFD library was constructed using the procedure of part III.

Three different sequences were tracked for each of the six aircraft (Figs. 8-13). The images shown are of resolution 512 by 512, and the resolution of the first frame in each sequence is about 64 by 64, increasing to about 200 by 200. We consider this a quite demanding test of any shape analysis algorithm, since many subtle as well as overt problems can appear when unknown shape resolution differs from library shape resolution by this factor.

We would also mention in particular, the fact that the nearest library image to each frame is oriented randomly with respect to the unknown image, rather than at approximately the same angle. This has been shown to increase the difficulty of this type of classification and estimation. In an experiment in which random projections were classified, for example, 94.7% classification accuracy was obtained in a six class problem when the unknown data was oriented similarly to the library data. When an additional random rotation was added to the unknown data, classification accuracy dropped to 88%.

Tables 1 and 2 show the performance of the algorithm. Table 1 gives the number of frames identified correctly, and tracked correctly, out of 50. We say that the algorithm is tracking correctly if the angle error is less than .5 radian in both x and y. Since some of the non-uniformly spaced lines in the projection grid are separated by more than this angle (.67) we consider this a reasonable measure. Table 1 also gives the number of re-acquisitions required when the program decided that it had lost target lock. These of course slow down the algorithm significantly. It would perhaps be more realistic as well as more effective to skip a few frames of input when two or more of these occur in a row. This is probably due to bad data, normalization problems, or inadequate projection spacing.

Table 2 gives the average angle errors among the correctly tracked frames. For comparison purposes, the library covers a region of π by π radians with a grid of 13 by 11 projections. This means that the average sector widths are .262 and .314, for x and y respectively.

(5) Analysis of Performance and Computation Times

The programs used in the experiments of part V are written in Fortran, and are research tools rather than efficient implementations of the algorithms. They were run on a PDP-11/70 minicomputer, and the average time to track a sequence of 50 frames was 70.2 seconds. The bulk of this was spent in computing the NFD's, about 57.8 seconds. This computation included a convolution used in filtering the data, as well as the FFT. The computation can be reduced to 25 seconds for the 50 frames by using an array processor connected to the PDP-11/70. Thus we feel the potential for real time implementation using efficient programming is very high.

Tracking Algorithm Performance

	Sequence 1	Sequence 2	Sequence 3
Mirage	4,0,3	0,6,0	0,0,1
Mig	0,0,0	0,15,0	0,1,0
Phantom	8,0,6	0,0,0	0,0,0
F104	43,0,4	0,0,0	0,0,1
F105	1,1,7	0,12,0	0,6,1
B57	0,2,1	0,6,0	0,9,3

Table 1. A,B,C indicates that A frames in the sequence were identified incorrectly, B frames were tracked incorrectly, (with error greater than .5 radians in x and/or y), and c target re-acquisitions were required. The total number of frames in each sequence was 50.

Estimation Performance

	Sequence 1	Sequence 2	Sequence 3
Mirage	.0535 .0624	.0396 .0446	.0686 .0384
Mig	.0847 .0719	.0456 .0462	.1086 .0518
Phantom	.0670 .0636	.0861 .0630	.0601 .0485
F104	.0563 .0815	.0903 .0440	.0757 .0479
F105	.1129 .0688	.0633 .0483	.0775 .0462
B57	.0690 .1301	.0582 .0493	.0934 .1015

Table 2. The average x and y angle errors among the frames correctly identified and correctly tracked are shown, in radians.

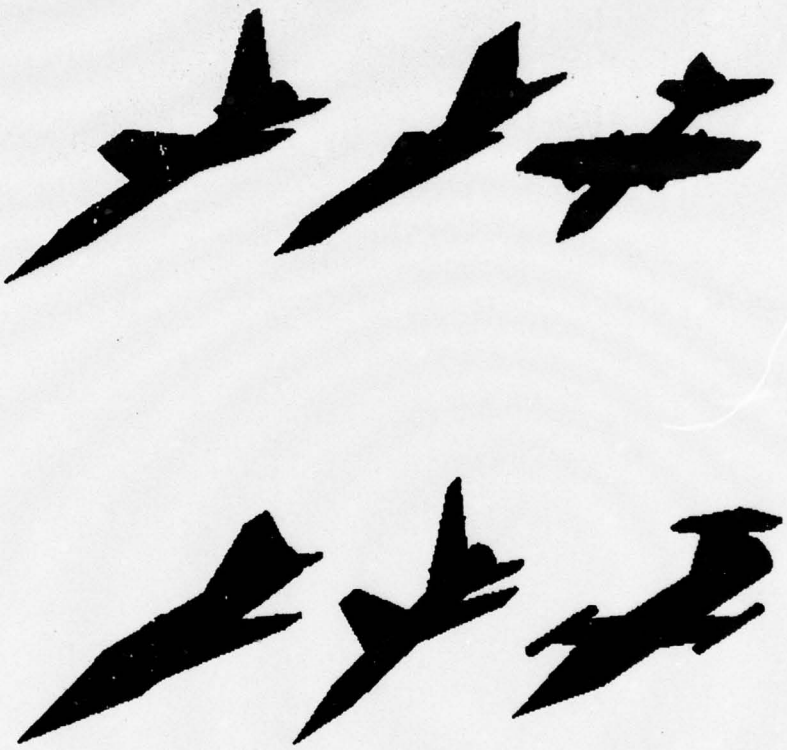


Fig. 1. Images representing each of the six 3D aircraft. The resolution is that of the library (256x256).

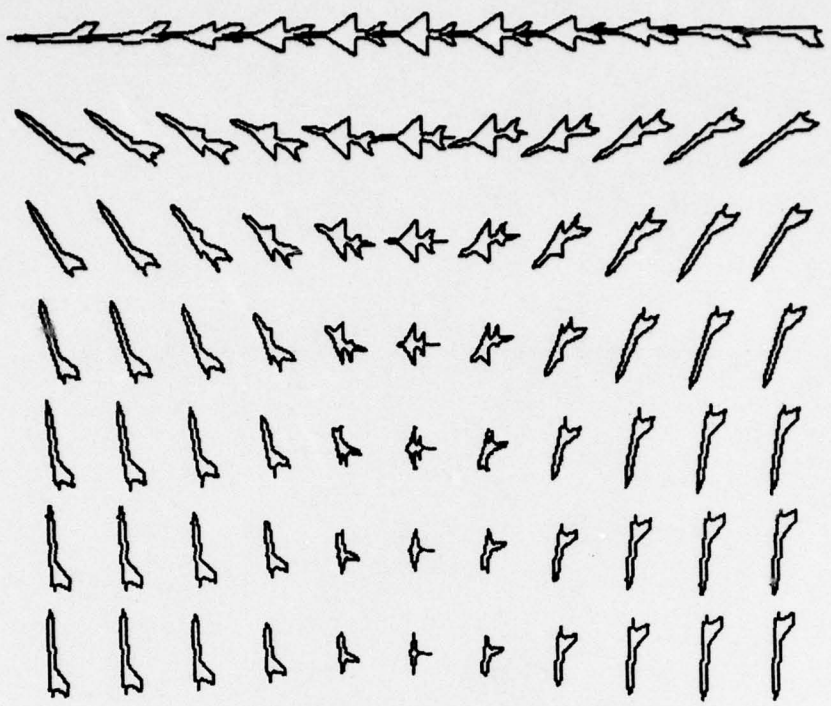


Fig. 3. One quadrant of the MiG projection library.

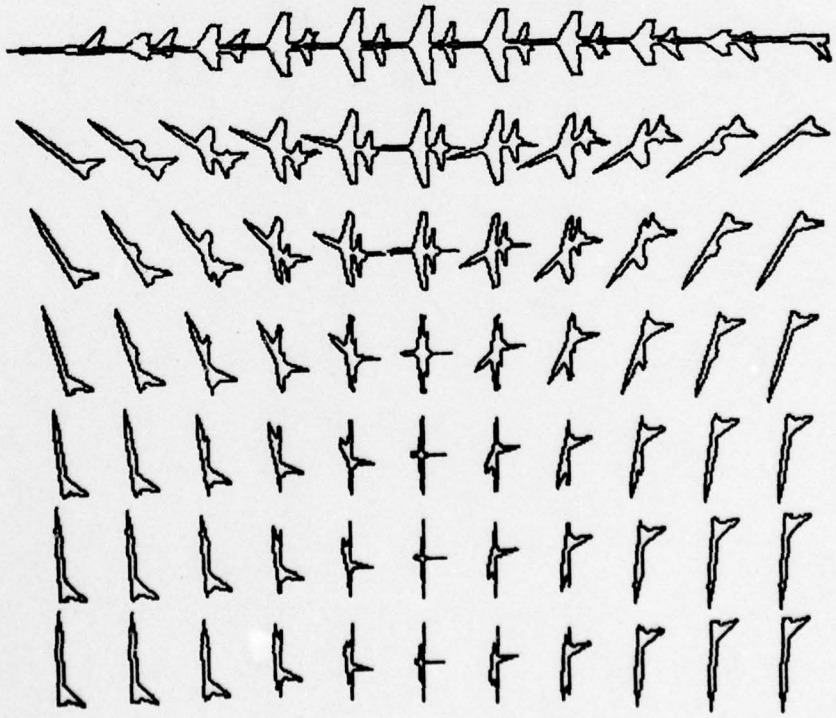


Fig. 2. One quadrant of the Mirage projection library.

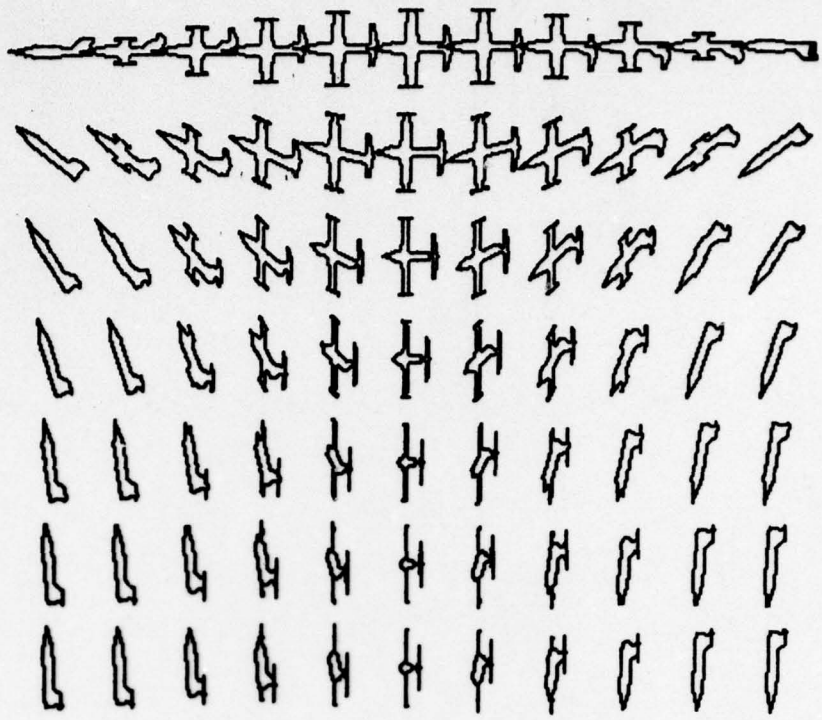


Fig. 5. One quadrant of the F104 projection library.

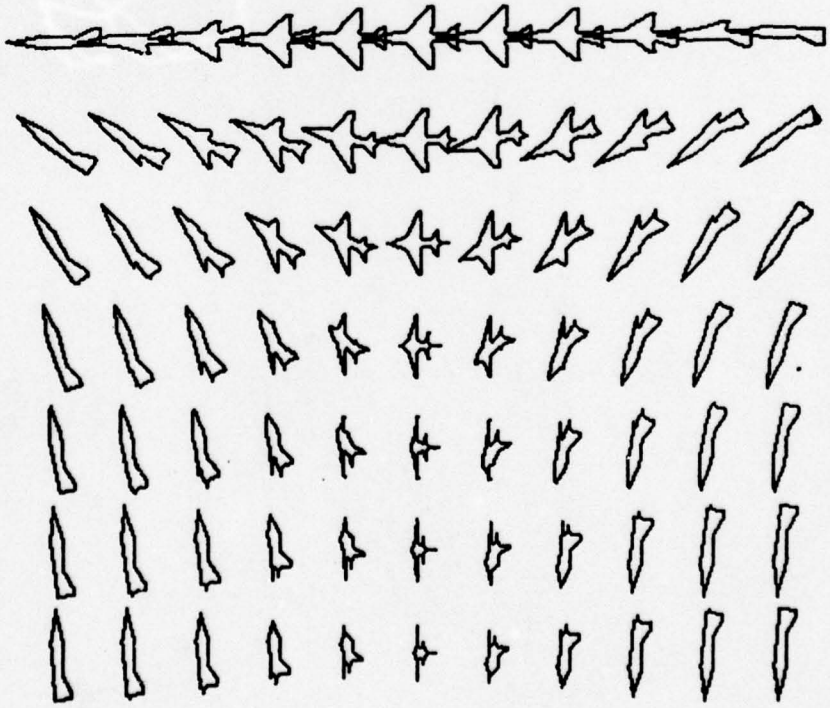


Fig. 4. One quadrant of the Phantom projection library.

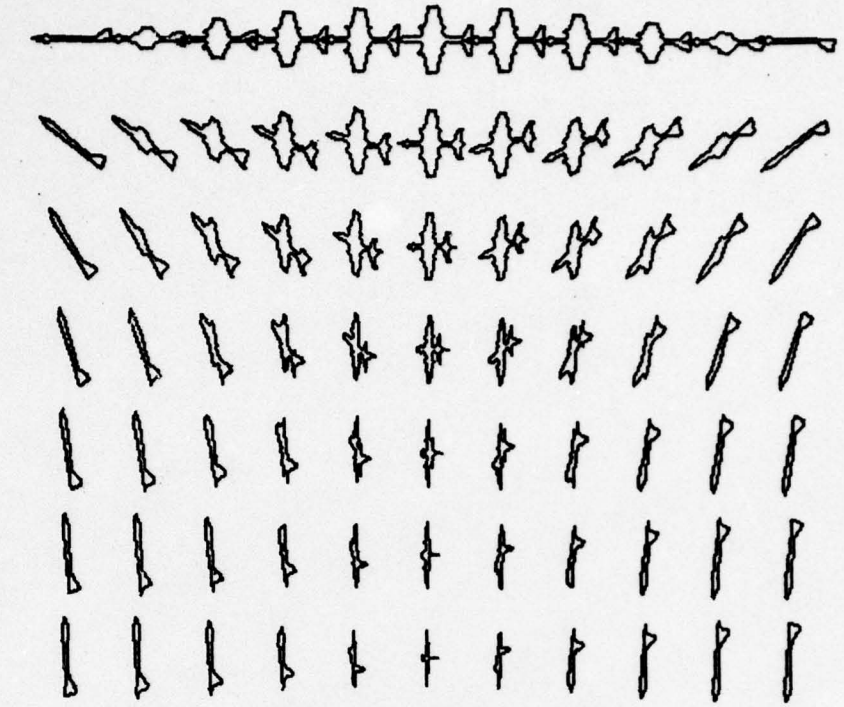


Fig. 7. One quadrant of the B57 projection library.

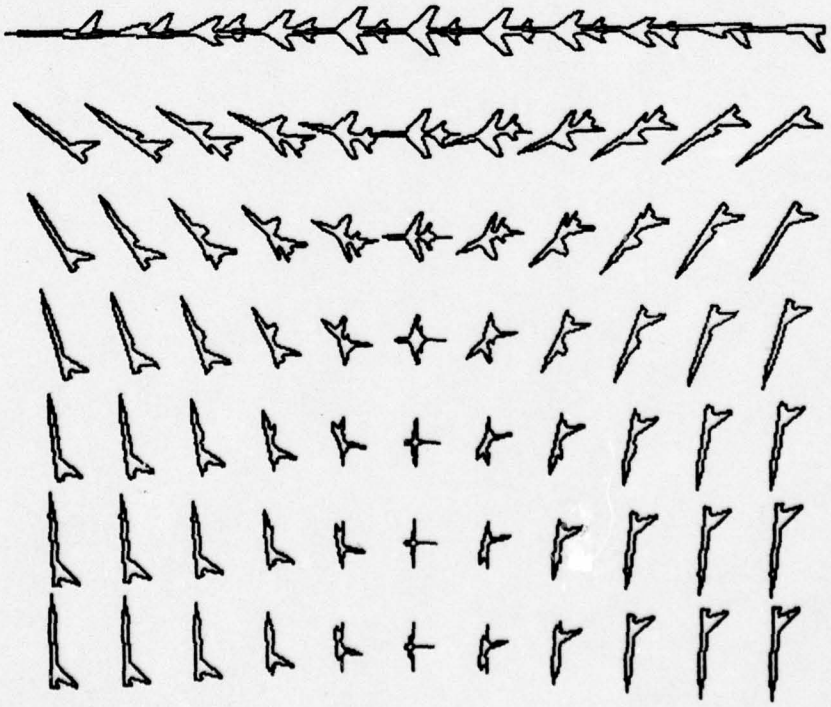


Fig. 6. One quadrant of the F105 projection library.

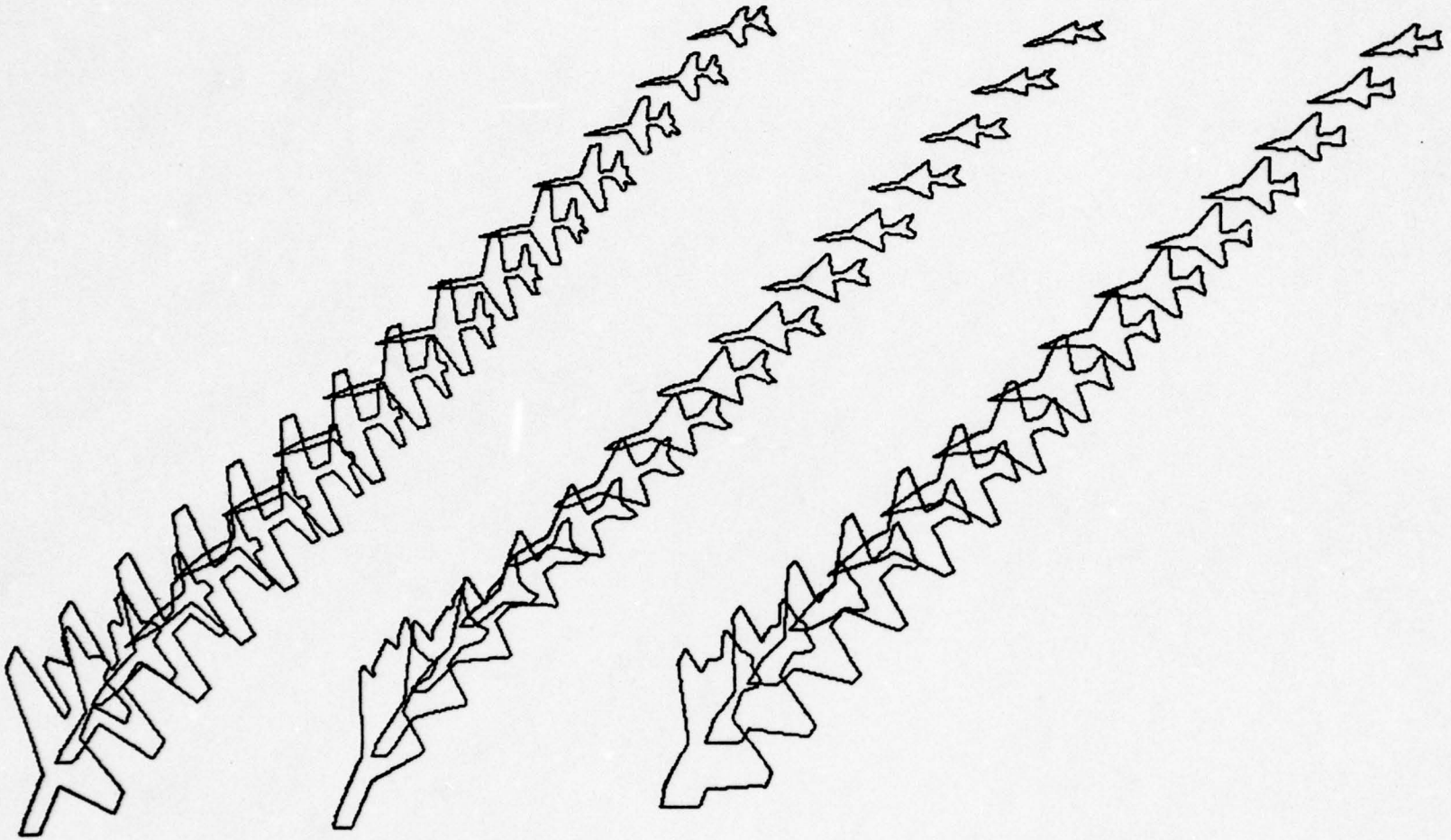


Fig. 8. Sequence 1: Mirage (left), Mig (center), Phantom (right).

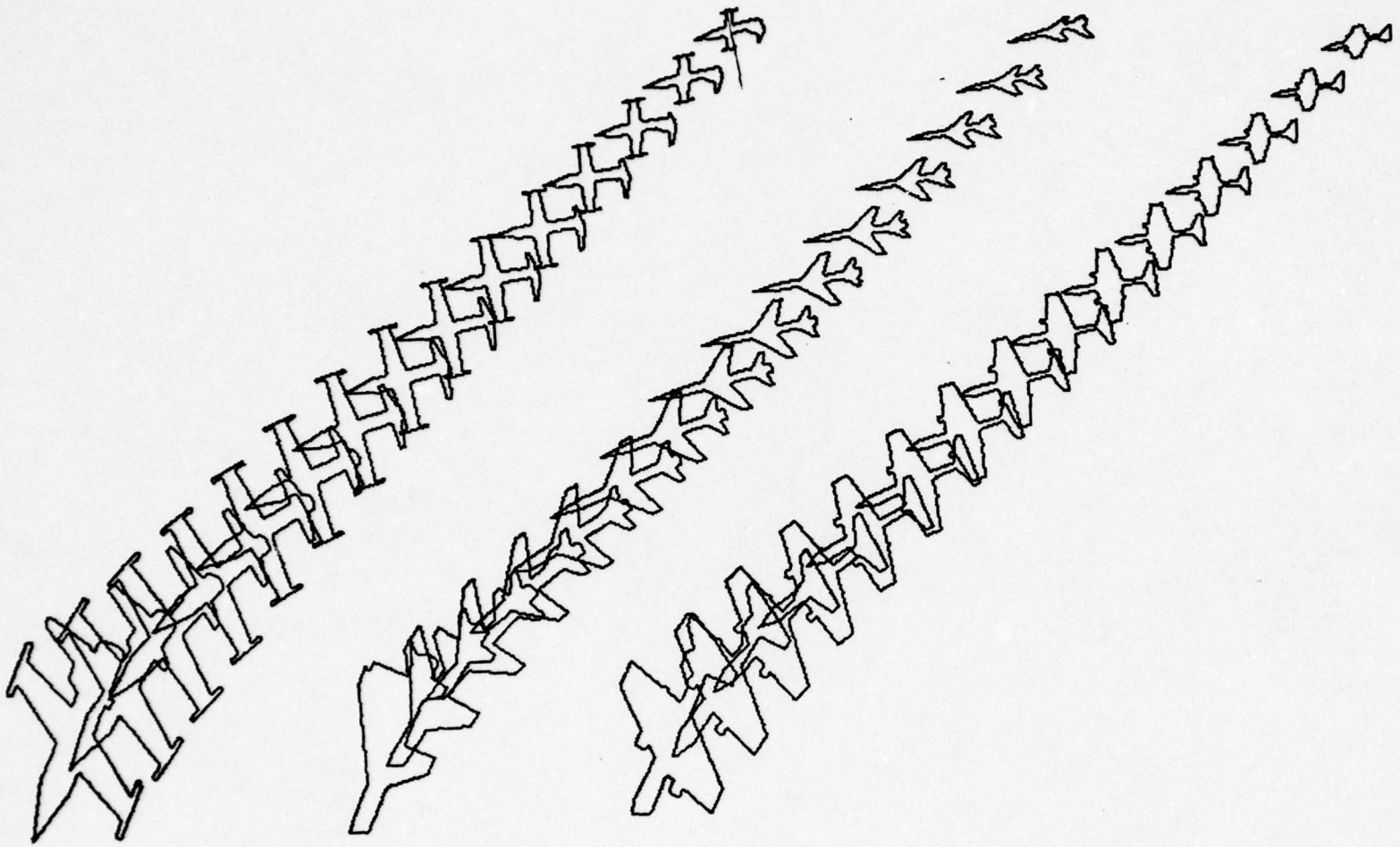


Fig. 9. Sequence 1: F104 (left), F105 (center), B57 (right).

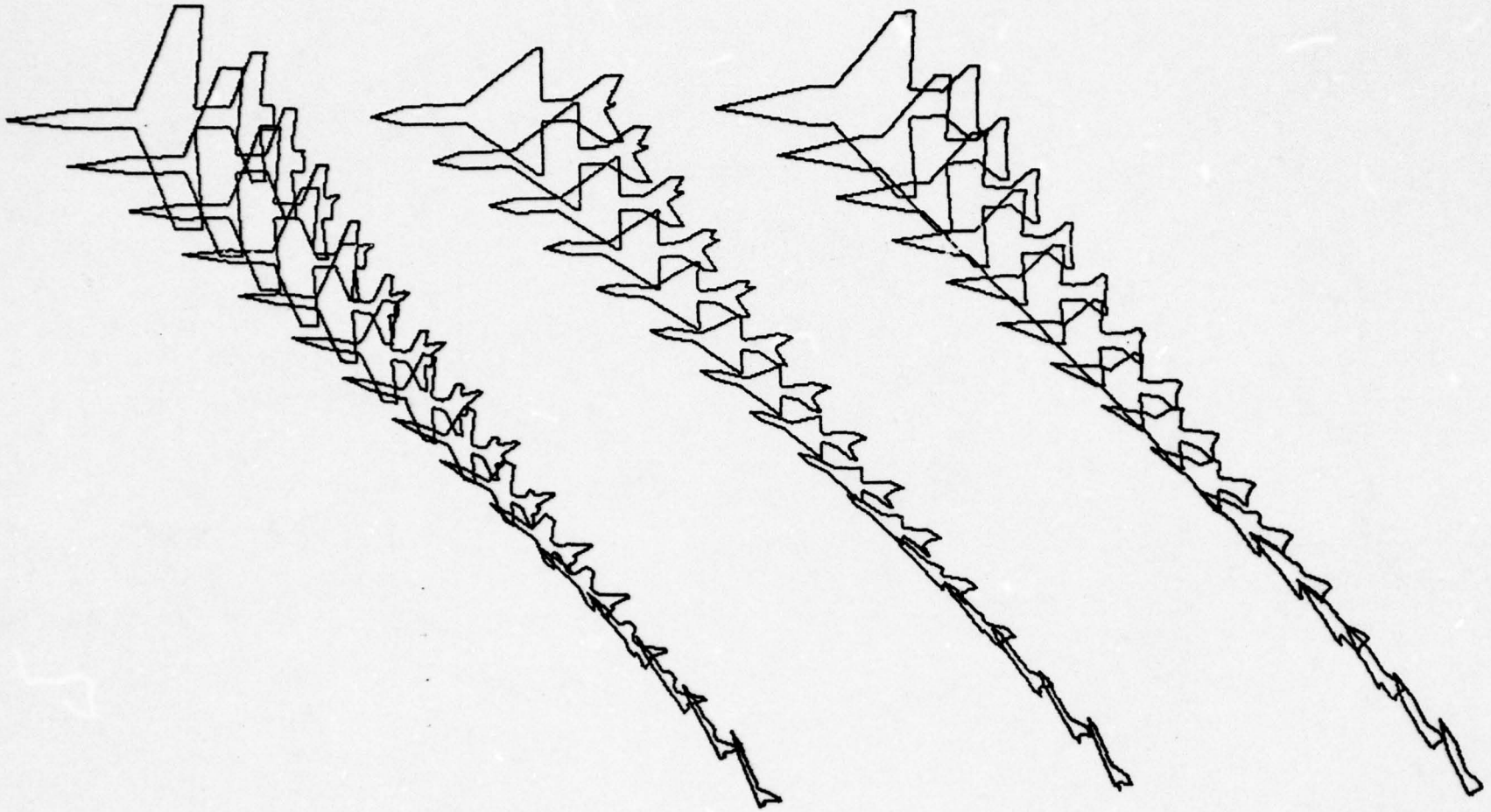


Fig. 10. Sequence 2: Mirage (left), Mig (center), Phantom (right).

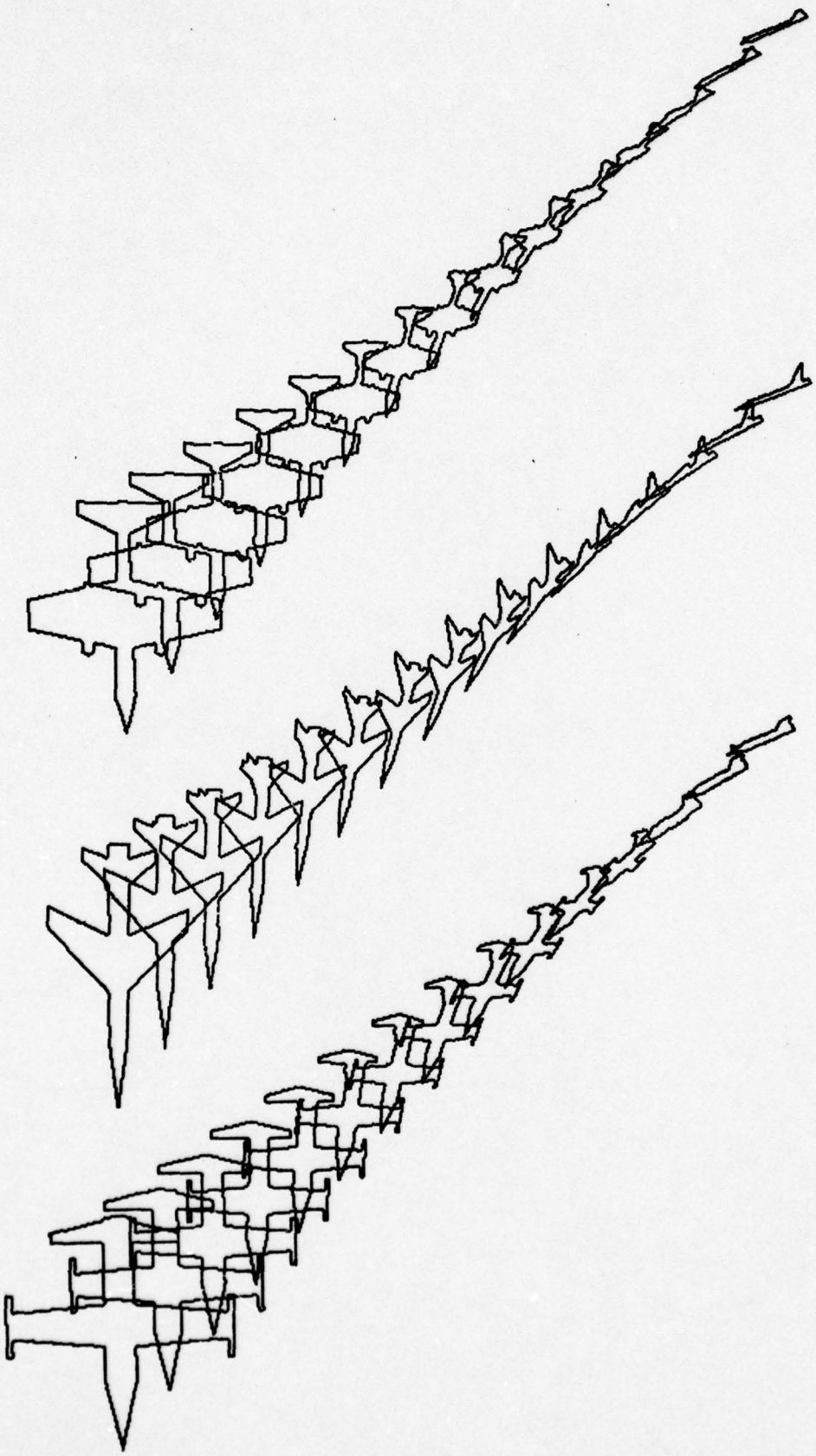


FIG. 11. Sequence 2: F104 (left), F105 (center), B57 (right).

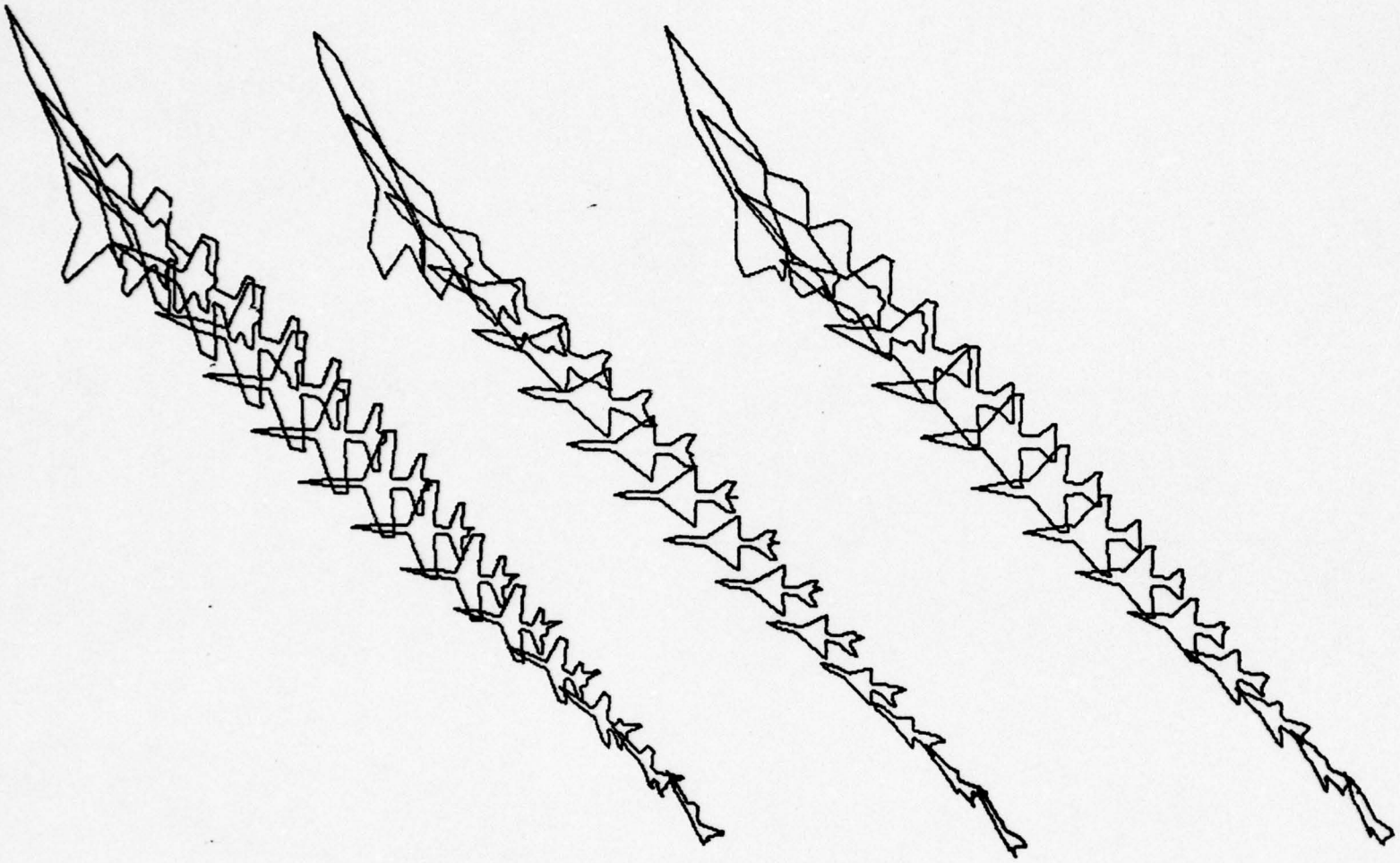


Fig. 12. Sequence 3: Mirage (left), Mig (center), Phantom (right).

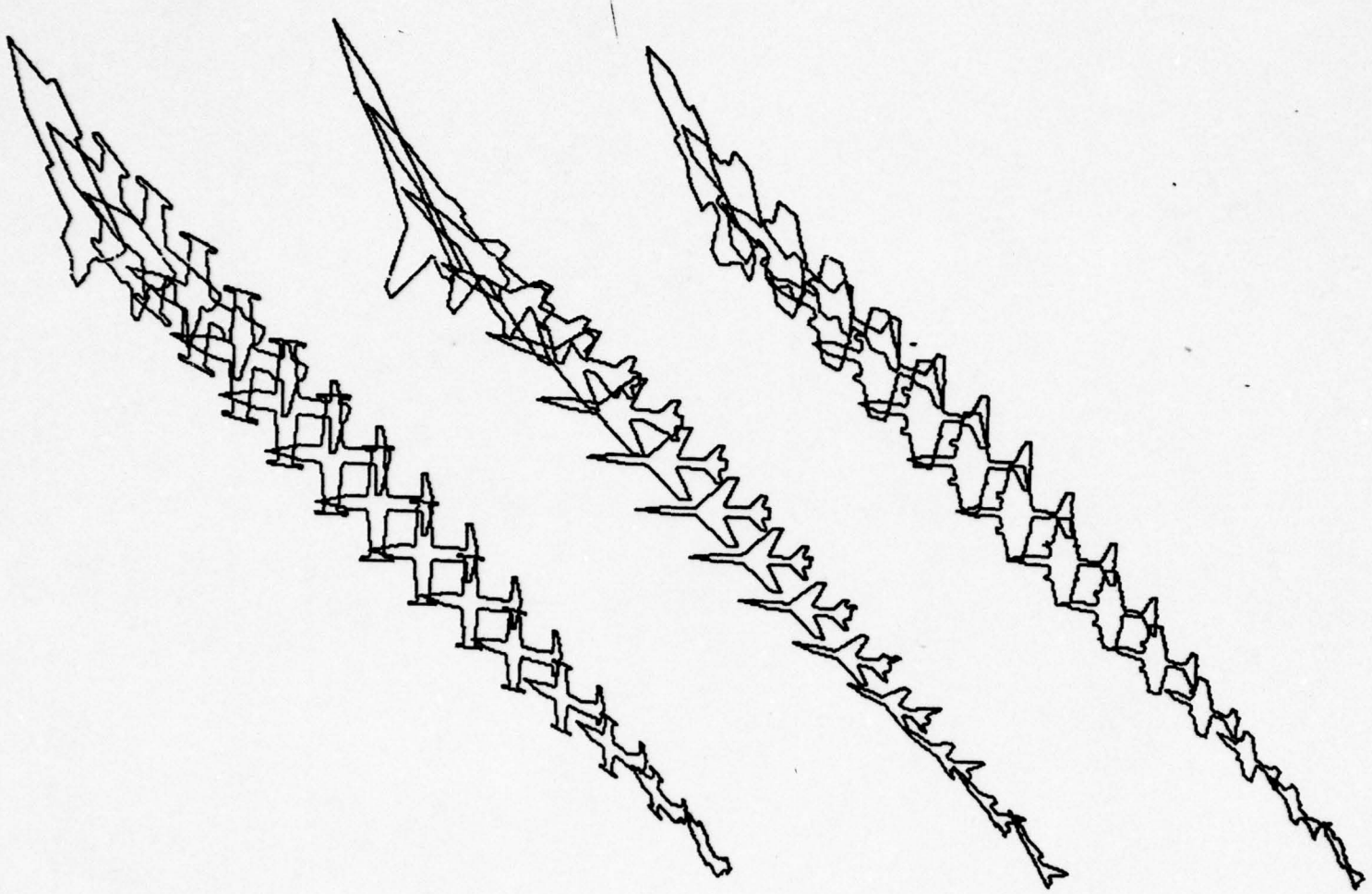


Fig. 13. Sequence 3: F104 (left), F105 (center), B57 (right).

B. Shape Analysis of Three-Dimension Objects Using Local Shape Descriptors.

This research has been documented as reference [2] in Section II and will be available in reprint form. Some of the major conclusions will be presented here.

Whenever part of a shape to be recognized is distorted or missing, global features such as Fourier descriptors are not useful. It is clear that if we want automatic machine recognition of shapes to rival the performance of human observers, we must provide some method of identifying partial shapes. Some form of local features must be used to accomplish this.

1. Overall Algorithm Description

The data input to this system consists of eight-neighbor chain codes describing the perimeter of the shape under analysis. A preprocessing routine is used to convert the input chain codes to more accurate x-y coordinate representations of each shape using a segmentation and curve-fitting procedure. This eliminates most of the chain code error.

After the preprocessing stage, the actual feature extraction algorithm derives the local feature vectors. The feature vector consists of alternate angles and distances. The angles represent total angular change about a curvature maximum, and the distances represent distances between curvature maxima. A dual segmentation is required to compute these features, in which both the curvature maxima, and intermediate curvature minima must be identified. The angle features consist of the total integrated angle between curvature minima, and the distance features consist of the total distance between curvature maxima.

The angle is clearly the best single feature, both from theoretical and experimental considerations. It is easy to compute, resolution invariant, and holds a lot of information. If a curvature minimum occurs near a region of slowly changing curvature, it can be difficult to reproduce its exact location for data of varying resolution, noise, etc. However, in this case, the angle is

very insensitive to the exact minimum location. If the curvature minimum occurs near a point of inflection, the angle exhibits substantial sensitivity to its exact location, but that location can be accurately and reproducibly determined.

The distances are measured between curvature maxima. These are quite well defined locations, so the distances tend to be fairly reproducible across various resolutions and orientations of data. In addition, the information contained in the distances is fairly independent of that contained in the angles. Finally, the distances work well in our feature vector comparison procedure, described below. The dependence between feature vector definition and the comparison procedures cannot be overlooked.

After computing the local feature vectors, the comparison algorithm compares each unknown vector to all of the library vectors, computing a distance between them where possible. The first step in this process is to find a good starting point; i.e. a local feature group in one vector which closely matches a local feature group in the other vector. Next, using this starting point, the program traverses the vectors simultaneously, combining local feature groups when necessary, in an attempt to derive two new feature vectors of identical dimension. If this fails, the structural analysis has indicated that the vectors are dissimilar. If it succeeds, a reasonable (statistical) distance measure is used to compare the two derived vectors.

2. Experimental Results.

A set of aircraft was synthesized using a graphics program. Three-dimensional models were constructed for six different aircraft, a Mirage, a Mig, a Phantom, an F104, an F105, and a B57. These models were then rotated through appropriate angles to create a library of projections. The projection space covered half of all the possible angles. If the object is imagined to be covered with a large hemisphere, the intersections of lines of latitude and longitude would represent projections.

The experiment consisted of classifying unknown, randomly selected projections by comparing their feature vectors to feature vectors computed from the library of projections. The unknown projections were of three different resolutions, 128x128, 64x64, and 32x32. Figs. 14-16 show representative unknown projections for each of the three resolutions.

Table 3 shows the performance of both the local and FD algorithms on data of three resolutions, with two different degrees of generality. Data (1) has unknown projections oriented similarly to the library set, and data (2) has an additional random rotation. It is clear that the local method is superior to the global method in classification accuracy for both the 128x128 and 64x64 unknown data. The 32x32 data showed a significant drop in performance, with the FD method being somewhat better. This is to be expected since the local method is more sensitive to the detailed behavior of the contour boundary than global methods.

It was somewhat surprising to us to find that this local method could beat a global method at its own game, classifying entire shapes. We realized that the general global method had little room for improvement, and could not easily take advantage of apriori information. In contrast, there is abundant opportunity to be clever with local features, by doing such things as looking for particular local structures known to be associated with objects of interest, and performing the classification based only on this information. We believed however, that if general methods of classifying shapes of complete objects were compared the global methods would prove quite powerful.

The explanation for this situation is perhaps not as difficult as might first appear. The Fourier series coefficients used as features in the FD procedure are all affected by any slight difference in shape. A short burst of noise which might occur over a small part of the contour can seriously affect every FD coefficient. On the other hand, our local feature can actually swallow up such a noise burst in

a combined feature, paying a penalty in the distance measure which may not be significant. Even if no combinations are involved, the distance measure weights every local feature by its total length, so a short feature can have small effect on the total distance.

The relationships between FD coefficients do not appear to point to an obvious distance measure for classification purposes. Most theoretical work on FD distances has relied on a mean square distance, but experimental results have shown that an absolute value distance measure is slightly superior in practice! No rigorous theoretical explanation for this has materialized, but one possibility is that the larger FD coefficients may be unduly affected by small variations in the contour being represented, and hence should not be weighted as heavily.

An attempt was made to improve the results of the FD method by sacrificing computation time. When the library shapes differ significantly from the unknown shape, which can happen with this low projection density, there is a small potential for normalization problems in the FD algorithm. Checking distances to all potential normalizations improved the performance of the FD method by an average of 1.5% while slowing it down by a factor of four to five. This improvement does not change the conclusions above. Using the preprocessing algorithm on the data did not improve performance.

The estimation performance of the FD method (Table 4) is clearly better than the local method, due to the estimation procedure described briefly above. The local method simply assumes that the unknown projection is oriented the same as the nearest library projection.

The FD method was much faster than the local method, as would be expected. Specifically, the FD program can compute a normalized FD in about .9 seconds, and perform a three-dimensional classification in about 1.0 seconds on a PDP 11/70 minicomputer. The corresponding times for the local feature program are 3.6 seconds and 20 seconds. Both of these programs are research tools rather than efficient implementations of the algorithms.

Classification Accuracy

Data Type	Local Method	FD Method
128x128 (1)	96.7 %	94.7 %
128x128 (2)	95.3 %	88.0 %
128x128 (3)	94.3 %	84.7 %
64x64 (1)	91.7 %	89.0 %
64x64 (2)	90.0 %	85.3 %
32x32 (1)	66.0 %	71.0 %
32x32 (2)	59.7 %	64.0 %

Table 3. Classification accuracy in the six class 3D experiment. Data is identified by resolution, and by type, with (1) being data of similar orientation to library data, and (2) being completely general data given an extra rotation. Although both algorithms are designed to recognize projections viewed only from above, data (3) shows performance on a set of 600 unknown projections, half viewed from above and half from below. Data (3) has the extra rotation of data (2).

Projection Angle Estimation Error

Data Type	Local Method		FD Method	
	x	y	x	y
128x128 (1)	.0728	.0825	.0543	.0315
128x128 (2)	.0790	.0834	.0556	.0478

Table 4. The 300 unknown projections are classified, and their orientation is estimated by both the local method and the FD method. The resulting median angle errors, in radians, are shown. The FD method is able to interpolate between library projections, while the local method simply takes the orientation to be that of the nearest library projection.

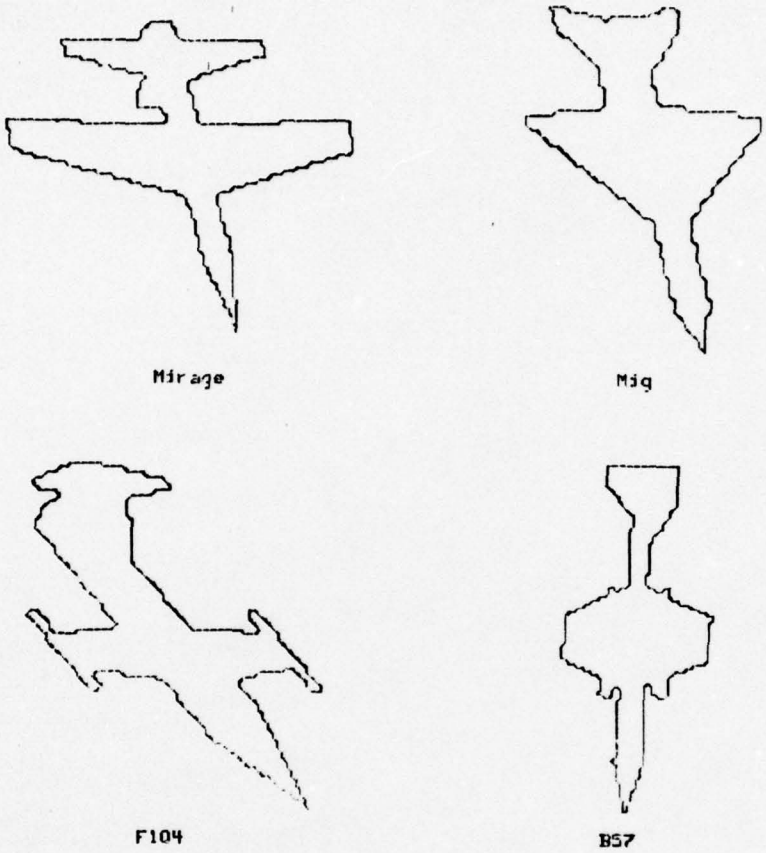


Fig. 14 Representative Unknown Contours (128x128).

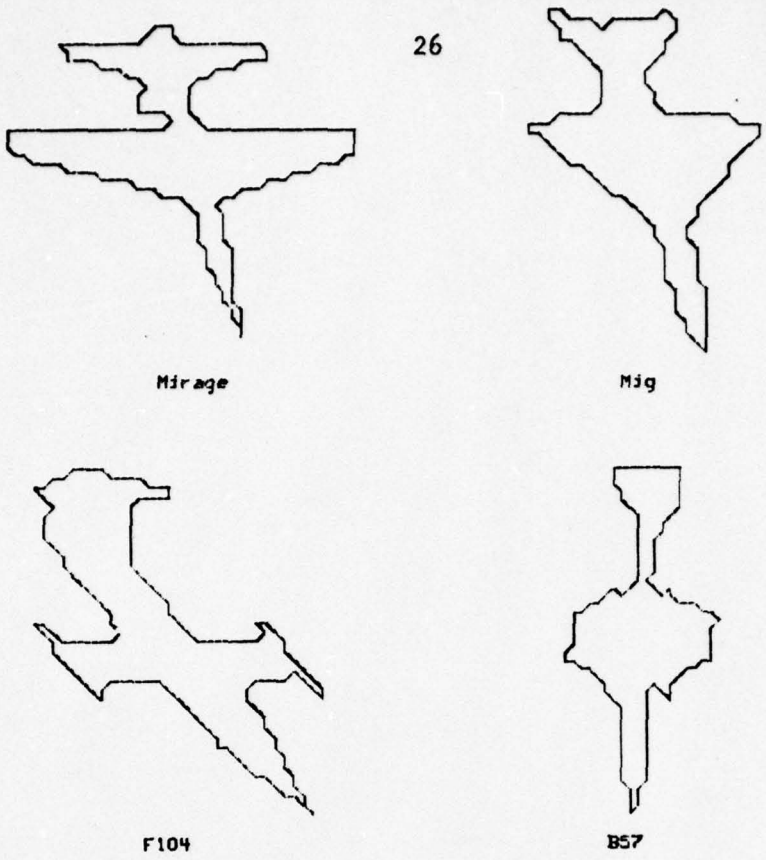


Fig. 15 Representative Unknown Contours (64x64).

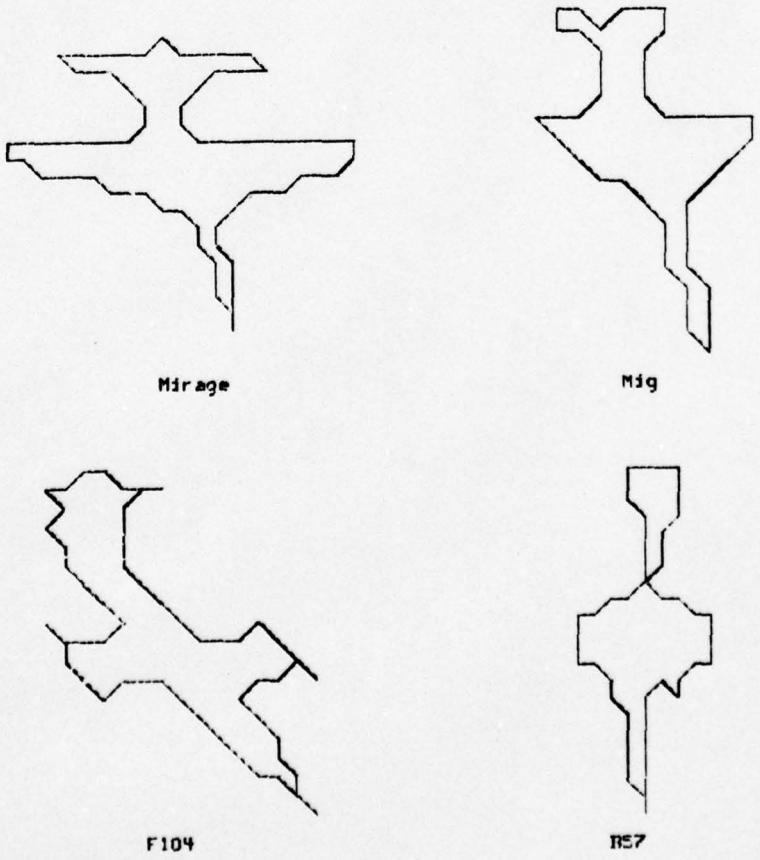


Fig 16 Representative Unknown Contours (32x32).

C. Sequential Image Analysis Using a Parallel Binary Array Processor

Reproduced on the following pages is reference [3] in Section II. References, Tables, and Figures refer to those on pages 39-44.

ABSTRACT

The application of a Binary Array Processor (BAP) to the rapid analysis of a sequence of images has been studied. Several algorithms have been developed which may be implemented on many parallel processing organizations. The characteristic operations of a BAP are discussed and analyzed. A set of heuristic algorithms are described which are designed to register two images of TV type video data in real-time. These algorithms illustrate the characteristic features of a BAP and their cost is analyzed in detail. The result of applying these algorithms to FLIR data and to noisy optical data are given. Analysis of these algorithms illustrate the importance of an efficient feature extraction hardware for image understanding applications.

Introduction

Binary Array Processors are SIMD processors consisting of a matrix of processing elements (PE's). Typically a BAP contains a large number of very simple PE's; each PE is capable of manipulating only a few bits of data with each instruction and arithmetic operations are achieved with bit-serial algorithms.

BAP's have been developed over the last 20 years mainly for image processing applications. Early BAP implementations were limited to in the order of 1000 PE's [1,2,3], however, with the advent of LSI technology, current BAP implementations consist of in the order of 10000 PE's. Most current PE design [4,5,6] are directed towards LSI implementation; aiming at 1 to 8 PE's with their data memory on a single chip.

The important features of the BAP organization are: fast parallel operation, fast access to local neighborhood data and efficient storage of the image data. Algorithms developed for efficient implementation on a BAP are also well structured for other bit-serial processors such as STARAN [11] and, in many cases could be implemented on other parallel processors such as Illiac IV [12].

BAP's are restricted to 1-bit wide data paths in general. With this constraint it is possible to calculate the minimum number of memory cycles needed for many characteristic operations. From these figures the cost of algorithms may be calculated without being restricted to a single PE architecture. The ratio of cost between different operations are more marked for BAP than for a conventional computer. For example, typically a logical operation may be 10 times faster than an add operation and 100 times faster than a multiplication. Some practical heuristic image processing algorithms are presented, which are matched to the efficient operations of a BAP. The

general approach is to reduce the data to a binary image form which may be very rapidly manipulated on a BAP.

A set of algorithms have been developed for the rapid registration of two images from a sequence. These are suitable for real time implementation on a BAP for TV video data. The sequence of algorithm is as follows.

(a) Median Filter: The images are median filtered to remove spurious large valued noise pixels.

(b) Adaptive Threshold: The filtered image is thresholded into a binary feature image by a locally adaptive algorithm.

(c) Balanced Binary Filter: An iterative filter is applied to the binary feature image to remove isolated noise pixels.

(d) Binary Image Registration: One binary image is slid over the other until the best match is found.

In the next section, the characteristics of the BAP architecture are discussed and memory cycle times for some operations are given. Each of the image processing algorithms are then described in detail. Finally, the results of applying these algorithms to some real data are given.

Binary Array Processors

The main functional units which characterize most BAP PE architectures are shown in Fig. 1. All data paths are 1-bit wide and data is stored in a 1-bit wide memory M. Data selection is achieved by the input select unit which can obtain data from the memory M or from near neighbor PE's. Data processing is achieved with a simple ALU which can process two single-bit operands per instruction. The input select unit and ALU may contain several 1-bit registers for holding temporary data values. Some PE designs [1,5] involve a 1-bit mask register which, when set, inhibits the PE from processing data.

The data interconnections between PE's are shown in Fig. 2. Most PE architectures [1-5,7] have interconnections with the 4 nearest neighbor PE's. Some BAP designs [3,4,7] also have interconnections with the 4 next nearest neighbor PE's as indicated by the broken lines in Fig. 2. Hexagonal PE interconnections have also been considered [3,7-9]. The input select unit can select data from any memory location within its own PE or any adjacent-connected-PE memory location.

The performance of a BAP for several functions on N-bit integer data is given in Table 1. A memory cycle is the basic instruction time for all PE instructions, even if the instruction does not access the data memory M. The memory cycle times are optimal if the BAP is restricted to 1 bit wide data paths and a single data memory M. Only a simple 2-input ALU is necessary to achieve these optimal speeds. Multiplication may be achieved in $3N^2$ cycles and division may be achieved in $6N^2$ cycles by algorithms using the operations defined in Table 1. However much higher speeds for multiplication and division are possible if a more complex ALU with increased temporary storage is available.

A matrix of PE's as described above, is capable of realizing low level image processing functions such as filtering and edge detection. For image understanding, a feature extraction mechanism is necessary, two possible functions are given in Table 2. The first function determines if any bit in a bit-plane is set. This is easy to implement and only requires one memory cycle. The second function computes the sum of a matrix of pixels. The implementation of this function requires some special hardware to sum data from the PE matrix; the execution time depends upon the complexity of this hardware.

The execution for a typical mechanism which shifts data out through one edge of the PE matrix is given in Table 2. A value for d of 1 is easily attainable; moreover, smaller values for d may be achieved with additional hardware.

Median Filtering

The adaptive threshold algorithm is very sensitive to any large valued spurious noise pixels. This type of noise can be effectively removed by a 3x3 local area median filter. The true median of the 9 element area may be computed on a SIMD processor by a sequence of 20 compare and exchange operations [10]. For a BAP processor this would require 122N memory cycles if all 8 near neighbor processors are available and 124 N memory cycles if only 4 adjacent PE's are available.

A pseudo median filter operation has been developed which requires fewer memory cycles to implement than the true median function. This algorithm may select one value higher or lower than the true median value but it is still adequate for removing random high level noise pixels. The pseudo median is formed in two stages: First the median of each horizontal triplet of pixels values is computed, i.e.

$$HM(i,j) \leftarrow \text{median}(P(i-1,j),P(i,j),P(i+1,j))$$

where HM is the horizontal median of the image P. At the same time the PE's above and below PE(i,j) compute HM(i,j+1) and HM(i,j-1). The second stage is to compute the median of each vertical triplet of the horizontal median filtered image, i.e.

$$PM(i,j) \leftarrow \text{median}(HM(i,j+1),HM(i,j),HM(i,j-1)$$

On a SIMD processor a three element median may be achieved with 3 compare

and exchange operations; this implies 18 N memory cycles on a BAP (by substituting for the cost of operations defined in Table 1). Two of the exchange operations may be replaced by replace operations; this reduces the number of memory cycles required to 16 N. The algorithm for the median of three elements A, B, and C is given below

```
    M ← 1 iff A > B
T1,T2 ← exchange (A,B,M)
    M ← 1 iff C > T1
   T1 ← select (C,T1,M)
    M ← 1 iff T1 > T2
   MD ← select (T2,T1,M)
```

where M is a binary mask, T1 and T2 are temporary storage values and MD is the true median.

The pseudo median algorithm requires two three element median operations, i.e. 32 N memory cycles for a BAP.

Adaptive Thresholding

An adaptive thresholding algorithm assigns a 0 or a 1 to represent the value of each pixel position depending upon some locally computed statistic. One scheme, mean thresholding, is to represent a pixel by a 1 if it is greater than the local mean pixel value. An alternative scheme, which produces somewhat similar results, is to represent a pixel by a 1 if it is closer to the maximum local pixel value than the minimum local pixel value. This max-min thresholding is defined by

$$MMT(i,j) \leftarrow 1 \text{ iff } 2P(i,j) > Max_L(P(i,j)) + Min_L(P(i,j))$$

where Max_L and Min_L are the local maximum and minimum values computed over a local area of LxL pixels. The max-min scheme has produced better results

than mean thresholding for the data we have used.

The maximum of two values may be computed by a compare operation followed by a select operation. To compute the maximum of a row of M adjacent values only requires $\lceil \log_2 M \rceil$ compare and select operations. For example if, in one dimension, we have $Max_4(P(i))$ and we require $Max_8(P(i))$ only one compare and select operation is required as

$$Max_8(P(i)) = Max(Max_4(P(i-2)), Max_4(P(i+2))).$$

However, 8/2 shifts of the data are also required on a BAP to achieve this computation. In general for $Max_M(P(i))$ the cost of the computation is $\lceil \log_2(M) \rceil$ compare and select operations plus M-1 shift operations. For the two dimensional case, the maximum of each row is first computed and then the maximum of the resulting columns is computed; therefore twice as much computation is required. A similar algorithm can be use for comuting the local minimum.

The minimum number of memory cycles for computing the local maximum and minimum may be found by substituting the values for the operations given in Table 1. The number of memory cycles to compute $Max_L(P(i,j))$ is $2N(5\log_2(L)+L-1)$. A further slight reduction in this computation time is possible by observing that the data to be processed by a PE only has to be shifted to an adjacent PE rather than the PE itself; a saving in the order of $4N\log_2(L)$ shift operations is possible. The cost from $Max_L(P(i,j))$ then becomes $2N(3\log_2(L)+L)$.

Once Max_L and Min_L have been computed the remainder of the binary image computation requires one addition and one comparison i.e. (3N+1) + (2N+2+1) = 5N + 4 memory cycles. Therefore the total computation for the max-min adaptive threshold is $4N(3\log_2(M)+M) + 5N + 4$ memory cycles. The computa-

tion for the mean threshold function only requires $2N(\log_2(M)+M+1)$ memory cycles.

Balanced Binary Smoothing

The binary image which results from thresholding an image frequently contains regions of 'noise' caused by either indistinct edges or very fine detail in the original image. This noise may be removed by iteratively applying the balanced binary smoothing algorithm described below. This algorithm complements the values of any elements which have only one or less, 4-connected adjacent elements of the same value.

$$BB(i,j) \leftarrow 1 \text{ iff} (B(i,j) + B(i+1,j) + B(i-1,j) + B(i,j+1) + B(i,j-1)) \geq 3$$

where B is a binary image and BB is the smoothed version. This may be implemented on a BAP in 26 memory cycles with an optimum sequence of logical and sum operations. A single pass with this algorithm will remove any isolated 1's or 0's. Repeated applications of this algorithm will reduce any single bit wide lines of 1's or 0's, that are not connected to larger regions of the same type, by 1 bit with each application.

Binary Image Registration

A measure of dissimilarity or difference between two images f and g may be defined by the following

$$df = c \sum_{i=1}^{X} \sum_{j=1}^{Y} (f(i,j) - g(i,j))^2$$

where c is a constant.

Two images may be registered by sliding one image over the other and evaluating the difference at each shifted position until a minimum differ-

ence is found. As one image is sifted over the other some pixels near the edge of the image will not have any corresponding pixels in the other image; therefore these pixels must be discarded in computing the difference. The value of c is also affected when only a subset of the image is involved. One solution to this problem is to use a consistent subset of the image f(i,j) for all correlations. For a maximum displacement of r elements between the two images all elements of f(i,j) within r elements of the edge must be discounted from the correlation computations. A measure for relative difference may be defined as follows.

$$V(k,l) = \sum_{i=r+1}^{X-r} \sum_{j=r+1}^{Y-r} (f(i,j) - g(i+k,j+l))^2$$

where k is the horizontal displacement and l is the vertical displacement between g and f. c is a constant value for all V(k,l) therefore the minimum value of V indicates the best correlation.

The cost of the computation of this function on a BAP is very high. The squared difference between the images could require in the order of $3N^2$ operations with a simple squaring algorithm and the cost of extracting the value of V from the result is 2NdX memory cycles (using the mechanism described in Table 2).

If the images are binary valued then the expression for V(k,l) reduces to

$$V(k,l) = \sum_{i=r+1}^{x-r} \sum_{j=r+1}^{j-r} f(i,j) \oplus g(i+k,j+l)$$

If the binary images are the result of the adaptive thresholding algorithm with an LxL window then the L/2 elements at the edge of the image do not

contain valid data. Therefore r in this case, is the maximum displacement between the image plus L/2.

The amount of computation required to compute the exclusive-OR function of the selected elements of f and g is 3 memory cycles. The extraction of V from the PE's requires Xd memory cycles. Our implementation of this function involves an equivalence function in place of the exclusive-OR function. This changes the registration problem to detecting the maximum of a peak rather than a minimum, the results, however, are identical.

Results

The algorithm described in the previous sections have been applied to two sets of test data with a BAP simulator [13]. In Fig. 3(a) two frames of sequential infrared images of a truck are given and in Fig. 4(a) two digitized sequential frames of a missile being launched, which were digitized from a 16 mm film, are shown. The truck images consist of 128x128x8-bit pixels and the missile images consist of 256x240x7-bit pixels.

The frames of the missile were pseudo median filtered and the filtered images are shown in Fig. 4(b). The frames of the truck were formed from two interlaced sensors with different sensitivities which resulted in alternatively pairs of scan lines with different intensites. To compensate for this imbalance the vertical three element median operation was replaced by computing the means of vertical adjacent pixel pairs. The result of the horizontal median followed by the vertical average operation is shown in Fig. 3(b).

All the filtered images were then max-min thresholded with a local window size of 15x15 pixels; the binary image results of this operations are shown in Fig. 3(c) and Fig. 4(c). The results of iteratively applying the balanced smoothing operation to these binary images are shown in Fig. 3(d)

and Fig. 4(d). For all images the smoothing algorithm terminated within 4 iterations. This algorithm had a more marked effect on the truck image data as it contained a larger amount of high frequency noise.

In Fig. 3(e) the inverted difference matrix V is given for displacements of upto 4 elements in all possible directions. This matrix has a clear peak in which the largest value 2.7% greater than the next highest value. This peak indicates the position of the best correlation, i.e. 2 pixels to the right and 2 pixels down. The equivalence function of the two binary images with this displacement is shown in Fig. 3(f); the light areas indicate where the two binary images have the same value.

In Fig. 4(e) the inverted difference V for displacements of upto of 10 pixels in all possible directions is given. This matrix has a main peak value located at 3 pixels to the left and 8 pixels down. There is also a second peak located at 4 pixels up and 1 pixel to the right. This peak is due to the matching of the missile which has changed position, relative to the background, between the two frames. The maximum of the second peak is within one to two pixel positions of the registration of the missile. As the missile is only 2-3 pixels wide this is not close enough for accurate target registration, it is sufficiently close, however, to direct the camera to track the target.

In Fig. 4(g) the absolute difference between the two pseudo median filtered frames at the point of background registration is given; in Fig. 4(f) the equivalence function of the two binary images with the same displacement is given. The difference image clearly shows the changed area caused by the moving missile. The changed relative position of a bad vertical scratch in the 16 mm film can also be seen.

In Fig. 4(h) the difference matrix for the pseudo median filtered images is given. It has a very pronounced minimum at exactly the same position as the main peak maximum in Fig. 4(e). It is interesting to note that, in this case, there is no indication of the displacment of the missile with respect to the background.

The cost of realizing the algorithms on this data, assuming a BAP with as many PE's as pixels, is given in Table 3. The total cost for computing s difference values is as follows:

Truck: 128 d.s + 3s + 1164
Missile: 256 d.s + 3s + 1123

The value of s depends upon the amount of displacement between the two images and the efficiency of the search algorithm to locate the maximum value of the peaks. For typical values of s = 100 and d = 1 the truck image requires 14264 memory cycles to implement and the missile image requires 27023 memory cycles.

The difference matrix for the gray-level images requires a subtraction operation and a square operation for each difference point, moreover, the sum of the bits in 2N bit-planes needs to be computed. For a typical square algorithm involving $3N^2$ operations, the cost of computing the difference matrix for the missile data is:

3584 d.s + 199s + 224 .

Therefore the cost for s = 100 and d = 1 is 378524 memory cycles which is more than an order of magnitude larger than the binary feature image algorithms. A similar amount of time would be required by other similar classical measures such as convolution.

Conclusion

The memory cycle cost of a general BAP processor for some characteristic operations has been presented based on the 1-bit wide data-path constraint. The analysis of algorithms with this cost scheme indicates the efficiency of their implementation on many existing BAP architectures.

Algorithms for the efficient registration of a sequence of images on a BAP have been presented. These algorithms are more than an order of magnitude faster than classical measures. The difference matrices generated by these algorithm indicate where the backgrounds of the images are registered and also provide information on the displacement of moving objects.

Close study of peak for the moving object in Fig. 4(a) indicates that a registration within 1-2 pixel positions is possible. This information may be used for tracking the object but is not sufficient for registering the object (which is only 2-3 pixels wide). Current research is directed towards extracting the general shape of moving objects from background registered images and then using this information to more accurately register and analyze the object.

Finally, the importance of an efficient hardware feature extraction mechanism for implementing image understanding algorithms on a BAP is illustrated. For a value of d = 1, which implies an efficient counter at one edge of the BAP, feature extraction operations are responsible for over 90% of the total computation time of the registration algorithm.

References

1. Gregory, J. and McReynolds, R., "The SOLOMAN Computer," IEEE Trans. on Computers, December 1963, pp. 774-781.

2. McCormick, B. H., "The Illinois Pattern Recognition Computer ILLIAC III," IEEE Trans. on Computers, December 1963, pp. 791-813.

3. Duff, M. J. B., Watson, D. M., and Deutsch, "A Parallel Computer for Array Processing," Information Processing - North-Holland Publishing Company, 1974, pp. 94-97.

4. Duff, M. J. B., "CLIP 4: A Large Scale Integrated Circuit Array Parallel Processor," 3rd International Joint Conference on Pattern Recognition, 1976, pp. 728-732.

5. Fung, L., "A Massively Parallel Processing Computer," Symposium on High Speed Computer and Algorithm Organization, April 13-15, 1977. Also in "High Speed Computer and Algorithm Organization" (Kuck, Lawrie, and Someh Eds.), Academic Press, 1977.

6. Consini, P. et al., "The Serial Microprocessor Array (SMA): Microprogramming and Application Examples," The 5th Annual Symposium on Computer Architecture, April 3-5, 1978.

7. Rindfuss, R. and Reeves, A. P., "Hardware Design of the BASE 8 Binary Array Processor," Purdue Technical Report TR-EE 78-46, 1978.

8. Reeves, A. P., "A Systematically Designed Binary Array Processor," Submitted for publication to the IEEE Trans. on Computers.

9. Preston, K., "Feature Extraction by Golay Hexagonal Transforms," IEEE Trans. on Computers, September 1971, pp. 1007-1014.

10. Reeves, A. P., "Parallel Algorithms for Fast Median Filtering and Pseudo Median Filtering," to be published.

11. Batcher, K. E., "STARAN Parallel Processor System Hardware," National Computer Conference, 1974, pp. 405-410.

12. Barns, G. H. et al., "The Illiac IV Computer," IEEE Trans. on Computers, August 1968, pp. 746-757.

13. Reeves, A. P., "An Array Processing System with a Fortan Based Realization," accepted for publication in Computer Graphics and Image Processing.

Function		Memory Cycles
k-PE shift	$R(i,j) \leftarrow A(i,j+k)$ or $R(i,j) \leftarrow A(i+k,j)$	(k+1)N
add/subtract	$R(i,j) \leftarrow A(i,j)+B(i,j)$	3N
add and k-PE shift	$R(i,j) \leftarrow A(i,j)+B(i+k,j)$	(k+2)N
compare	$R(i,j) \leftarrow 1$ iff $A(i,j)>B(i,j)$	2N+1
Select	$R(i,j) \leftarrow (Ai,j)xM(i,j)+B(i,j)xM(i,j)$	3N
Exchange	$R1(i,j) \leftarrow A(i,j)xM(i,j)+B(i,j)x\overline{M}(i,j)$ $R2(i,j) \leftarrow A(i,j)x\overline{M}(i,j)+B(i,j)xM(i,j)$	4N

PE-Function Execution Times

Table 1

Function		Memory Cycles
detect any PE with a 1	$V \leftarrow \bigvee_{i,j} A(i,j)$	1
sum all elements	$V \leftarrow \sum_{i,j} A(i,j)$	dxXxN

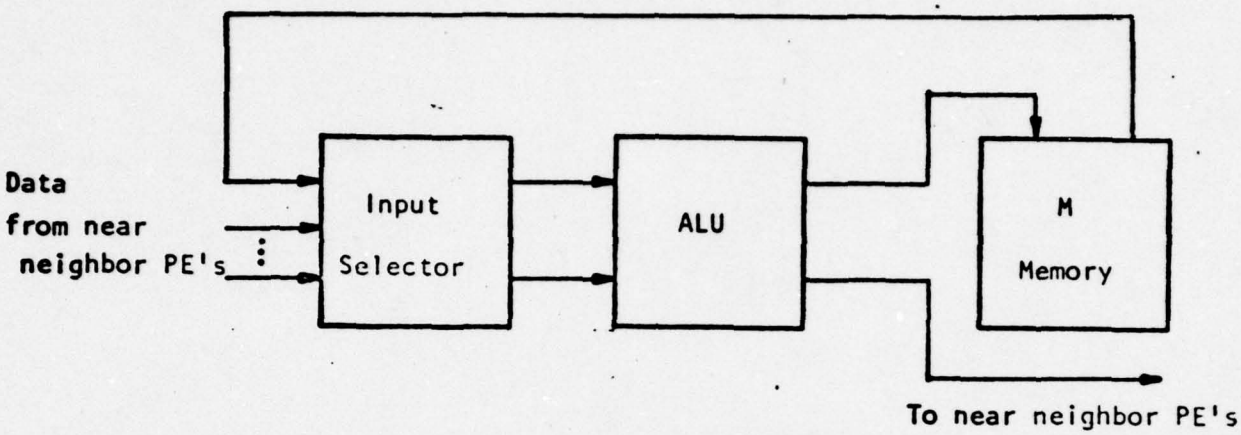
Feature Extraction Times for a BAP with XxY PE's

Table 2

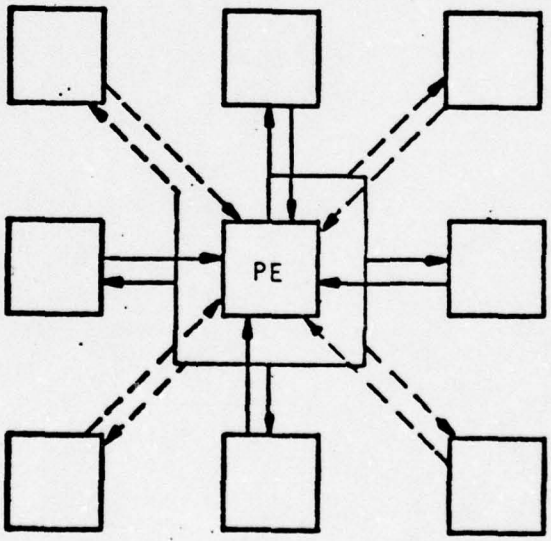
	Truck	Missile
dimensions of image	128x128x8	256x240x7
Pseudo median filter	152	224
Max-min threshold on 15x15 area	908	795
Balanced smoothing, 4 iterations	104	104
Total for binary image generation	1164	1123
Binary difference measure per point	128d+3	256d+3

Table 3

Cost of Registration Algorithm (Figures for All Algorithms are in Memory Cycles).



BAP PE Organization
Figure 1



Data connections between a PE
and its near neighbor PE's
Figure 2

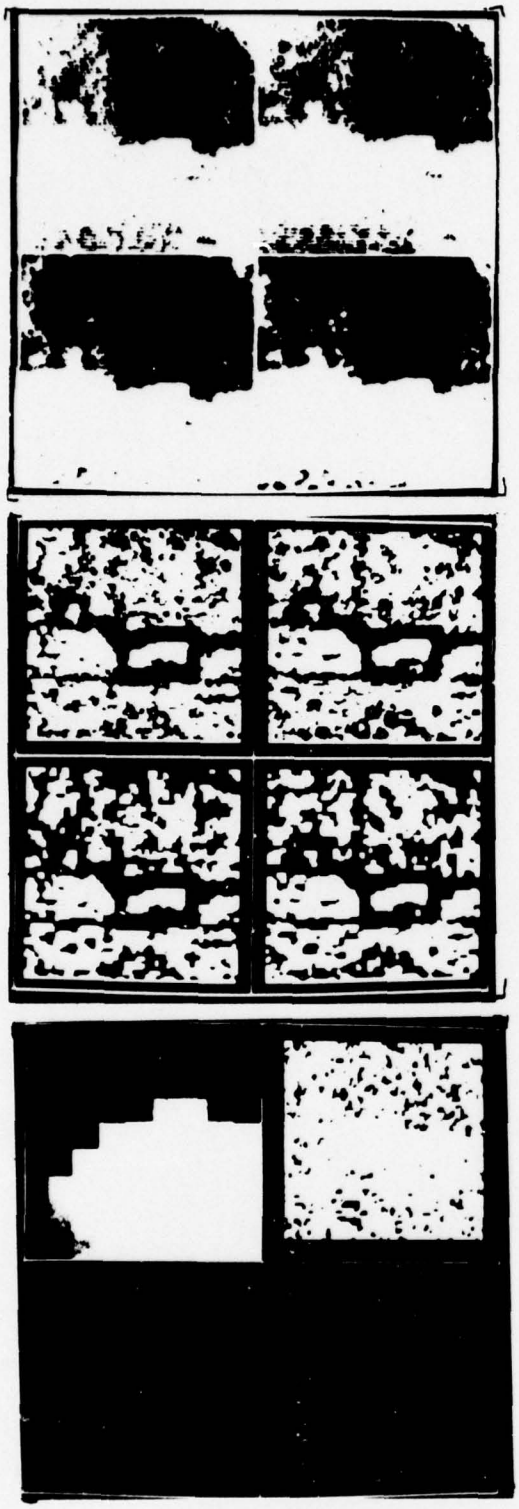


Figure 3
Two frames of FLIR data

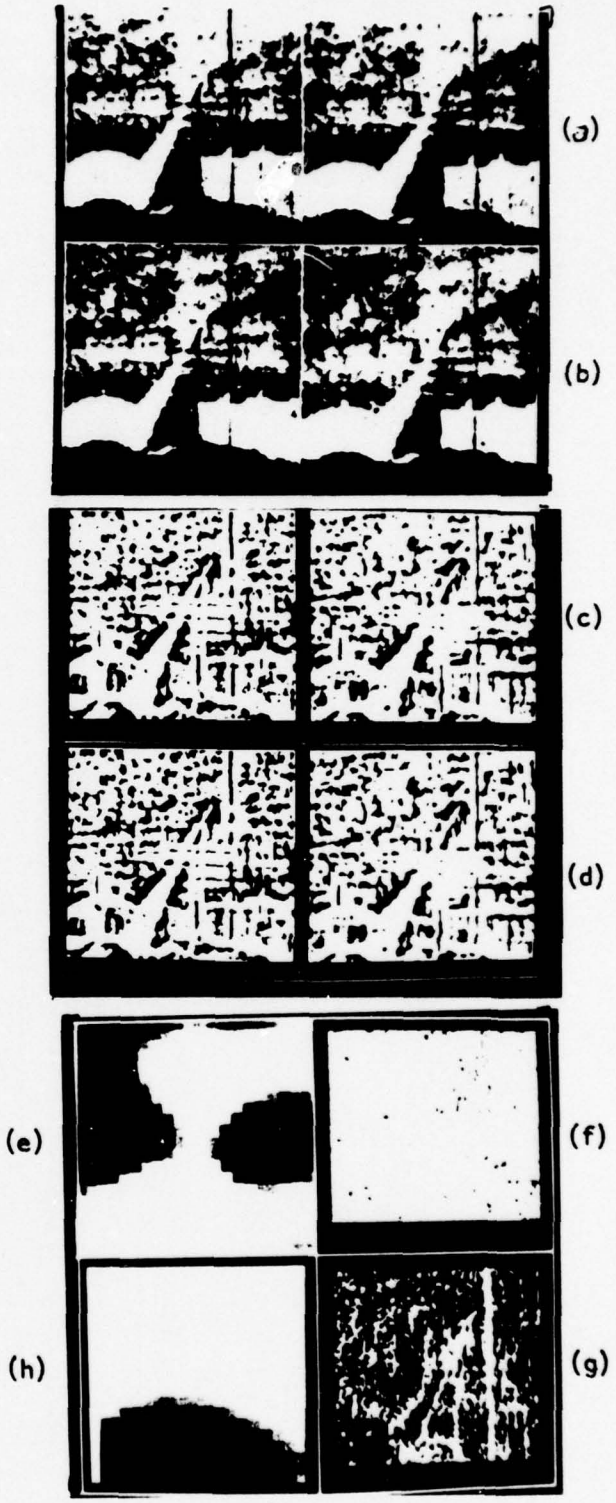


Figure 4
Two frames of a moving missile

(a) The original images
(b) Median filtered images
(c) Adaptive threshold with a 15x15 window
(d) Balanced smoothed images
(e) Inverted difference matrix
(f) Match between registered binary images
(g) Absolute difference between registered images
(h) Difference matrix of the grey level images

D. The Use of Edge Information in Segmentation and Tracking

The following results are similar to but newer than those reported in Reference [4], Section II. This has not yet been submitted for publication. References and Figures refer to those on pages 54-63.

ABSTRACT

A method of averaging is described which uses edge information to control the window shape, thus allowing a large window to be used without the usual disadvantage of smearing the object with the background. A set of 3x3 and 5x5 isotropic weighting functions for edge detection are presented. A method using texture is described. Use of prior frame information in segmentation is demonstrated in a tracking application.

Index Terms - edge detection, isotropic window, segmentation, texture, tracking.

I. Introduction

Automatic video tracking depends heavily on the segmentation of object from its background [1]. In many cases, the approximate location of the object to be tracked is known. Thus the problem is reduced to object extraction and verification followed by camera keying to track the desired object.

The method of segmentation and tracking presented here is greatly dependent upon edge detection. The method is relatively insensitive to slight edge discontinuities. Best segmentation, however, requires good edge extraction.

II. Edge Operators

Much work has been done in the area of edge detection and several different approaches have been taken. It is generally recognized however, that boundary detection is best accomplished by the combination of a relatively simple edge detector, followed by algorithms that thin and link the segments obtained [2].

The proposed edge detector is a gradient type measurement made over a 3x3 window or a 5x5 window. Fig. 1 shows the weighting functions used for both size windows to yield isotropic detection. The gradient edge value at each point is the square root of the sum of a^2 and b^2, where a and b respectively, are the dot products of each weighting function with the same window size of image points centered around the point being determined. In order to simplify computations, the square root of the sum of squares is sometimes replaced with the sum absolute values. The 3x3 weighting functions yield the same result as functions previously published [2]. The second column of Fig. 2 shows the results applying the 5x5 edge operator to the original images shown in the first column. The original pictures are forward looking infrared (FLIR) imagery (thermal radiation) of four military vehicles.

Edge thinning is accomplished by hysteresis smoothing [3], [4] and local maxima extraction. Rosenfeld and Kak have previously described a non-maximum suppression method of edge thinning [1]. The algorithm used here retains only those edge points which are both (1) above a threshold, t_1, in edge value and (2) are hysteresis smoothed local maxima above another threshold, t_2. To determine the location of edge points the following procedure is implemented:

Compare the edge value of each point with the threshold t_1. Only those points above t_1 are considered. Each potential edge point is then compared with its two vertical neighbors. If its edge value is above both neighbors, the point is a local maximum in the vertical direction. If this is the case, compare this point with each point along both vertical directions until an edge value is encountered which is above the point's value or until the edge of a nxn window centered around the point in consideration is encountered. The largest differences between edge values in each vertical direction are then compared and the smallest of the two is retained as the size of the local maximum in the vertical direction. An example is shown in Table 1 for a 7x7 window.

Table 1. Sample Edge Values for Extrema Detection

32	36	40	47	30	24	20
31	33	34	30	32	36	32
34	36	40	32	40	30	28
38	42	46	45	43	35	33
34	36	40	33	47	32	30
30	34	42	50	42	40	38
26	30	30	20	45	36	34

The center value is 45. The 47 ends the search in the top direction and the 50 ends the search in the bottom direction. The range is 15 above (45-30) and 12 below (45-33). Therefore, the center point is a local maximum in the vertical direction of size 12.

This process is also done in the horizontal direction. In the example of Table 1, the center point is not a local maximum in the horizontal direction. If a point is a local maximum in both horizontal and vertical directions, the largest of the two sizes is retained. If the local maximum size is greater than t_2, that point location is marked as an edge point. The result of this process is a binary edge picture with thin edges. The result of this thinning process is illustrated in the third column of Fig. 2. The window size used was 21x21.

This edge selection method guarantees that there cannot be a square grouping of four edge points in the binary edge picture. As an example, we take four such points, a, b, c, and d as shown in Fig. 3. If $a > b$, then in order for b to be a maximum, we must have $b > d$. And likewise, $d > c$. But this gives $a > b > d > c$ so that c cannot be a maximum.

A problem with this method arises in the case of horizontal/vertical corners. As is apparent in the truck example of Fig. 2, the corner points are often neither horizontal or vertical maxima, and thus are destroyed by the thinning process. A solution is attempted by extending the local maximum detection method to use the two diagonal directions as well. The corners are improved, but the non-clustering guarantee described in the previous paragraph is lost. The fourth column of Fig. 2 shows these results.

III. Use of Edge Information in Averaging

Averaging is useful to the segmentation process in two major ways. First, it reduces noise present in the original picture, and second, texture information can be obtained by averaging local texture features derived from the original image [1], [5]. Fixed-window averaging also attenuates valuable high frequency information, blurring the resulting image. The following proposed technique greatly restricts this detrimental effect.

In a region where no edge points exist, grey level averaging is performed as usual over a square nxn window. However, where edge points are present, no averaging is allowed to cross an edge boundary so that each point is averaged only with those neighbors of its own region. This is implemented by allowing the predetermined edge boundaries to control the size and shape of the window. As illustrated in Fig. 4, searching is done in both horizontal and vertical directions. Searching is not allowed to continue beyond object edge or window boundary. Only those points which are encountered in both searches are used in the average. (Note: Since isolated edge points are meaningless they are removed prior to the averaging operation.)

This technique of edge-controlled averaging is illustrated in Figs. 5 and 6 and contrasted with the fixed-window method. In Fig. 5, the effects of a 7x7 averaging window on an original grey level picture are illustrated. Fig. 6 shows the results of a 15x15 window averager used on the texture features described in the next section.

IV. Segmentation

The techniques described here assume that the location of an object to be tracked is known. They attempt to separate the object and non-object points based on features measured in the background and in the object region. To accomplish object segmentation, background features are collected over an annular region surrounding the object. Then the features of the object region are compared to those of the background, the points not matching the background are labeled as object points. Grey level alone is not always enough information for accurate segmentation, so an additional feature is necessary for the process. The additional feature chosen to complement the grey level image is texture. The texture was chosen because it seems probable that object and background textures would not be identical, assuming a good texture measure were available to differentiate among textures. The texture feature used here is derived from hysteresis smoothed local extrema described elsewhere [6], [7]. The process is similar to that described in the previous section except that the original grey level image is used instead of the edge value image. Also minimum as well as maximum are retained. The size of the extrema retained is data dependent. The medium level extrema are retained and the low level extrema (noise) and high level extrema (grey level edges) are rejected. The resulting binary image is averaged to produce a texture value picture. This texture value corresponds to the number of medium level extrema in the vicinity of the point. The averaging process uses the edge information to control the window shape and size as discussed in the previous section.

The grey level image is also averaged over a small window using the edge information to produce a picture with more definition between the object and background. This is visible in the histograms shown in Fig. 7. The first histogram corresponds to grey levels in the first picture in Fig. 9. This original had only 40 grey levels which were multiplied by 4 to give the desired range. The histograms of the averaged picture using a fixed 3x3 window and using an edge-controlled 3x3 window are also shown. Note how edge-controlled averaging preserves or even enhances grey level differences while fixed window averaging blurs the results. Once the feature images are produced, two concentric circles are centered at the potential object location. The inner circle represents the potential object area and the annular region between the two circles represents the background region. In an automated system these circle sizes would be adaptive since approximate object size and background context will be available from prior frames. The background annular region must be large enough to allow a sufficient background sample to be collected but it must not include object points or be so large that irrelevant background obscures the background/object differences.

The background statistics gathering program generates a two-dimensional histogram over the two features for all background points. The quantization selected allows for 32 averaged grey levels and 16 texture values. This background histogram is therefore composed of 512 bins. Once the background 2-D histogram is completed, each potential object point (2-D vector) is com-

pared against its background bin. If that feature combination occurs often in the background, the point is considered another background point. If the feature combination does not occur in the background, that point is labeled an object point.

The results of applying this segmentation method to the originals in Fig. 2 is shown in Fig. 8. Included is the texture extrema, the texture value picture (edge controlled extrema averages), the edge-controlled average grey level, and the segmented results.

Following segmentation it is necessary to analyze the structure or the shape of the segmented object to verify that it is the object to be tracked. Three such possible methods of verification are the use of Fourier descriptors [8], moment and invariants [9], and projections [5]. These will not be treated in this paper.

V. Tracking

The method of the previous section was applied to sequential frames of a missile after take off. The results of this method are shown in Fig. 9. In each case, edges were detected using the 5x5 isotropic window. These edges were then used to average the original (128x128) over a 3x3 window and the texture extrema over a 15x15 window. These results were used as the two features for segmentation. Again it bears mentioning that the original data had only 40 grey levels.

In tracking problems, as opposed to single picture analyses, the segmentation can be improved by maintaining previous frame object histograms generated by the segmenter as described in section four. These prior statistics are multiplied by a scale factor less than unity and added to the new statistics. Or, in the case that present frame statistics yield a poor segmentation, (e.g. the object is temporarily hidden by clouds, etc.) prior

frame information may be relied upon entirely for the segmentation. This is possible in most tracking situations since the information contained in sequential frames can be assumed not to change drastically. This method of retaining prior frame information in segmentation was applied to the pictures of Fig. 9 and a comparison of the results is shown in columns one and two of Fig. 10. Column c of that figure contrasts the results of single frame segmentation of the same object without the use of edge information to control the averaging of the grey level and texture extrema.

VI. Conclusion

An intelligent use of edge information for segmentation and tracking has been proposed. Especially when texture is a significant factor in separating an object from its background, the edge information can be used to obtain much better texture results and therefore a better segmentation. On the limited set of data attempted for this paper, this method has shown improvement over similar methods where the edge information has not been used effectively.

Although the real time implementation of such a system has not been discussed in this paper, it is certainly a necessary step for practical use of the algorithms. One especially intriguing possibility is to implement the edge-controlled averaging is a two-dimensional analog CCD device, where the edge information is imposed as a potential that causes barriers in the conducting region, thus preventing averaging across edge boundaries. In fact, the strength of analog edge value might be used to control the strength of the barrier.

References

[1] A. Rosenfeld and A. C. Kak, Digital Picture Processing, Academic Press: New York, 2nd Ed. 1978.

[2] W. Frei and C. Chen, "Fast Boundary Detection: A Generalization and a New Algorithm," IEEE Transactions on Computers, C-26, October 1977, pp. 988-998.

[3] S. J. Mason and J. K. Clemens, "Character Recognition in an Experimental Reading Machine for the Blind," Recognizing Patterns, MA : MIT Press, 1968, pp. 156-167.

[4] R. A. Duda and P. E. Hart, Pattern Classification and Scene Analysis, John Wiley : New York, 1973.

[5] S. G. Carlton and O. R. Mitchell, "Object/Background Segmentation in FLIR Imagery," IEEE Computer Society Conference on Pattern Recognition and Image Processing, Chicago, May 31 to June 2, 1978, pp. 360-362.

[6] O. R. Mitchell, C. R. Myers, and W. A. Boyne, "A Max-Min Measure for Image Texture Analysis," IEEE Transactions on Computers, Vol. C-26, April 1977, pp. 387-391.

[7] S. G. Carlton and O. R. Mitchell, "Image Segmentation Using Textures and Grey Level," Proceedings of the IEEE Conference on Image Processing and Pattern Recognition: Troy, N.Y., June 6-8, 1977, pp. 387-391.

[8] E. Persoon and K. S. Fu, "Shape Discrimination Using Fourier Descriptors," IEEE Transactions on Syst., Man, Cybern., Vol. SMC-7, March 1977, pp. 170-179.

[9] S. A. Dudani et al, "Aircraft Identification by Moment Invariants," IEEE Transactions on Computers, Vol. C-26, pp. 39-46, January 1977.

$$\begin{matrix} \sqrt{2} & 1 & 0 \\ 1 & 0 & -1 \\ 0 & -1 & -\sqrt{2} \end{matrix} \qquad \begin{matrix} 0 & 1 & \sqrt{2} \\ -1 & 0 & 1 \\ -\sqrt{2} & -1 & 0 \end{matrix}$$

(a) 3x3 window

$$\begin{matrix} 1 & \sqrt{2} & \sqrt{3} & 0 & 0 \\ \sqrt{2} & 1 & \sqrt{2} & 0 & 0 \\ \sqrt{3} & \sqrt{2} & 0 & -\sqrt{2} & -\sqrt{3} \\ 0 & 0 & -\sqrt{2} & -1 & -\sqrt{2} \\ 0 & 0 & -\sqrt{3} & -\sqrt{2} & -1 \end{matrix} \qquad \begin{matrix} 0 & 0 & \sqrt{3} & \sqrt{2} & 1 \\ 0 & 0 & \sqrt{2} & 1 & \sqrt{2} \\ -\sqrt{3} & -\sqrt{2} & 0 & \sqrt{2} & \sqrt{3} \\ -\sqrt{2} & -1 & -\sqrt{2} & 0 & 0 \\ -1 & -\sqrt{2} & -\sqrt{3} & 0 & 0 \end{matrix}$$

(b) 5x5 window

Fig. 1 Isotropic weighting functions for edge detection

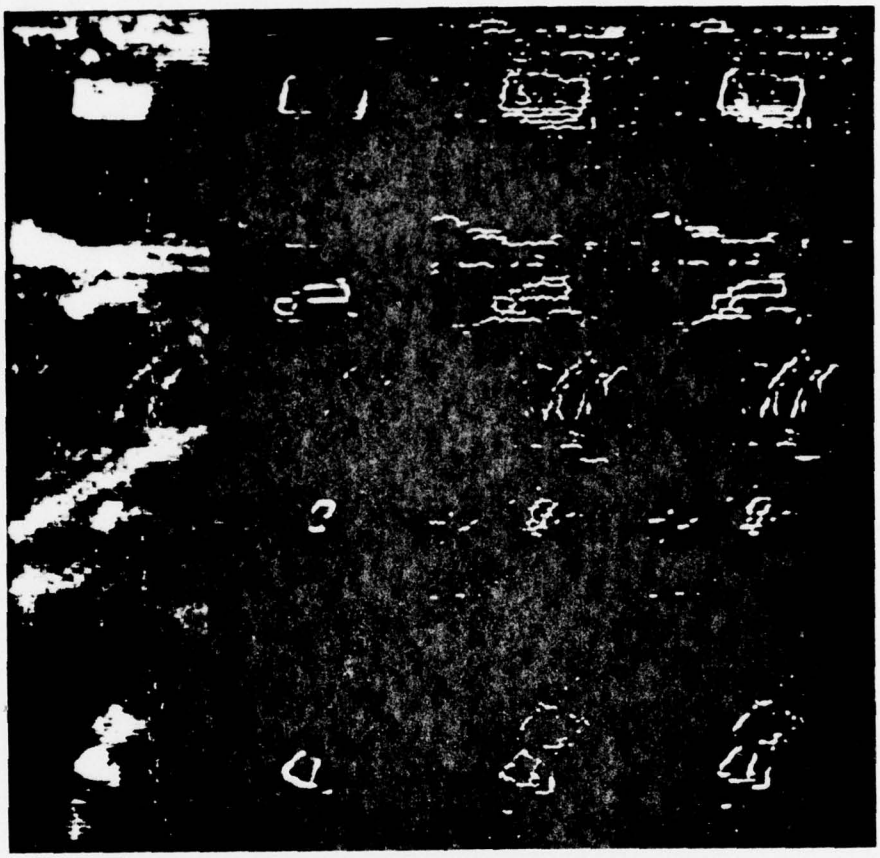


Fig. 2 Composite picture of sixteen 128x128 original FLIR images and results: (column 1) originals; (column 2) result of 5x5 isotropic edge detection; (column 3) bi-directional local maxima detection of column 2 picture; (column 4) quad-directional local maxima detection of column 2 pictures

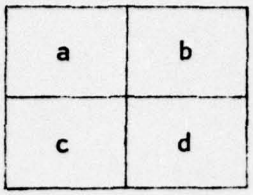
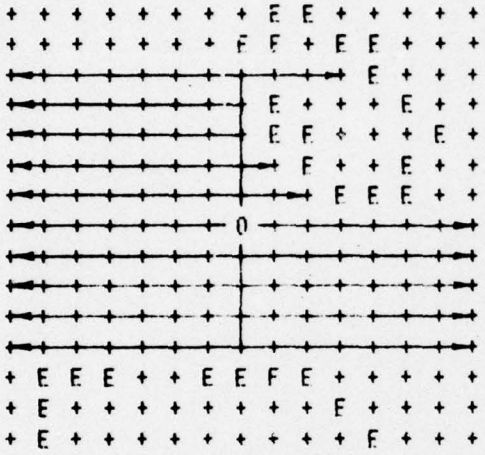
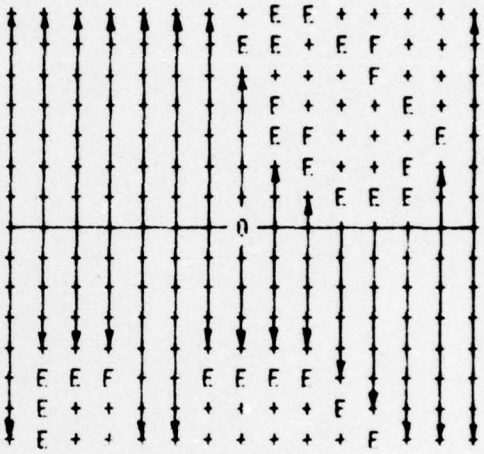


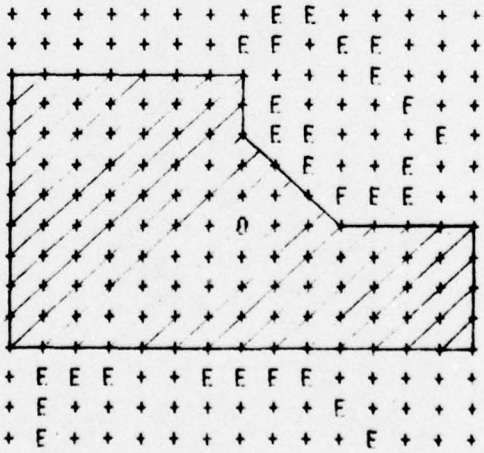
Fig. 3 Four adjacent edge values



Horizontal search
(a)



Vertical search
(b)



Points to be averaged
(c)

Fig. 4 Example of edge-controlled averaging

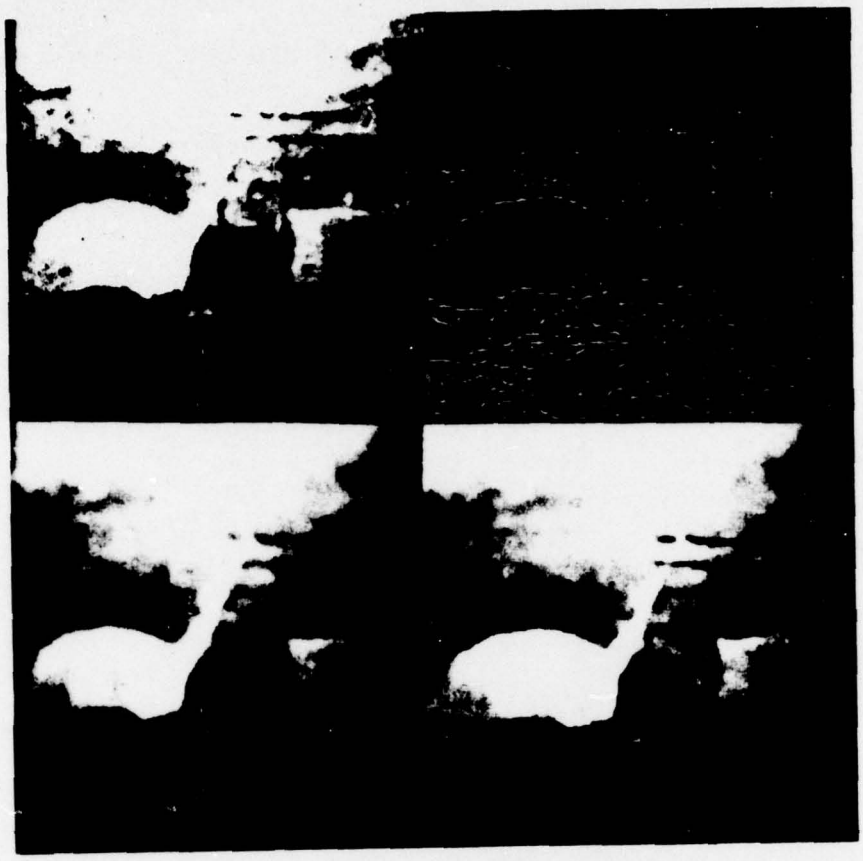


Fig. 5 Missile Launch: (top left) 256x256 original; (top right) Result of 5x5 isotropic edge detection; (bottom left) average over 7x7 fixed window; (bottom right) 7x7 edge-controlled average

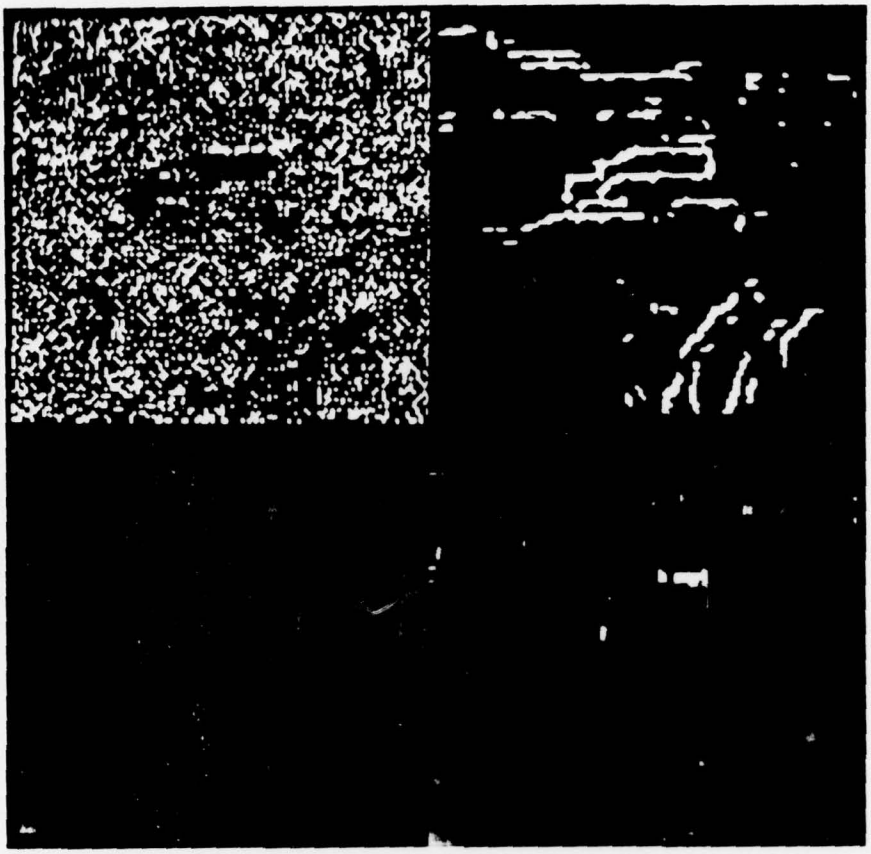
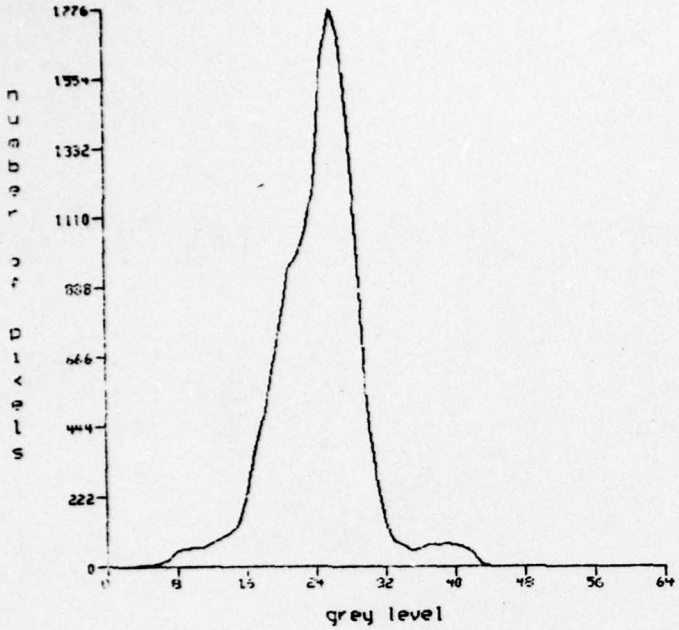
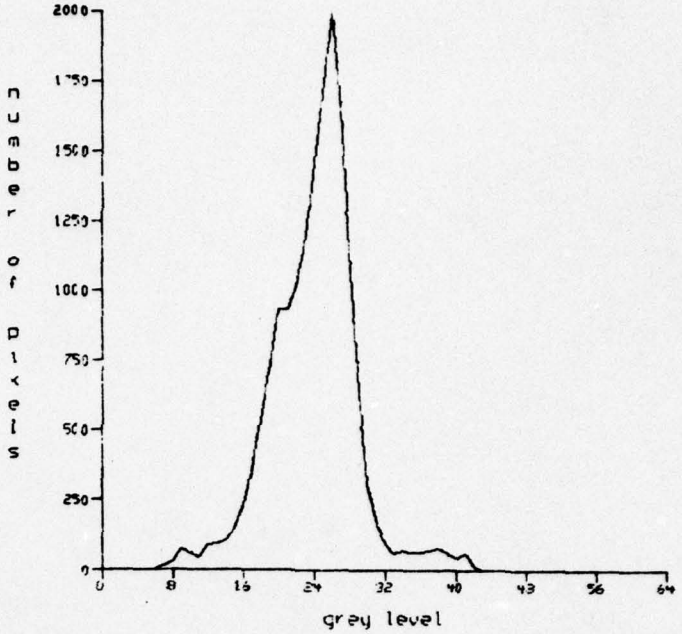


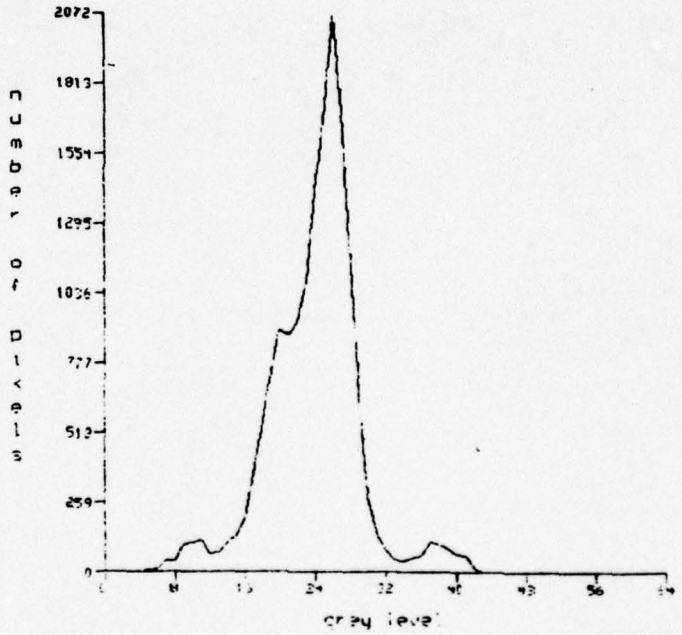
Fig. 6 FLIR truck of Fig. 2: (top left) 128x128 texture extrema; (top right) result of 5x5 isotropic edge detection on original image; (bottom left) average over 15x15 fixed window; (bottom right) 15x15 edge-controlled average



(a)



(b)



(c)

Fig. 7 Histogram correspo nding to the first picture of Fig. 9: (a) original grey level histogram; (b) histogram of averaged original using a 3x3 fixed window; (c) hostogram of averaged original using 3x3 edge-controlled window

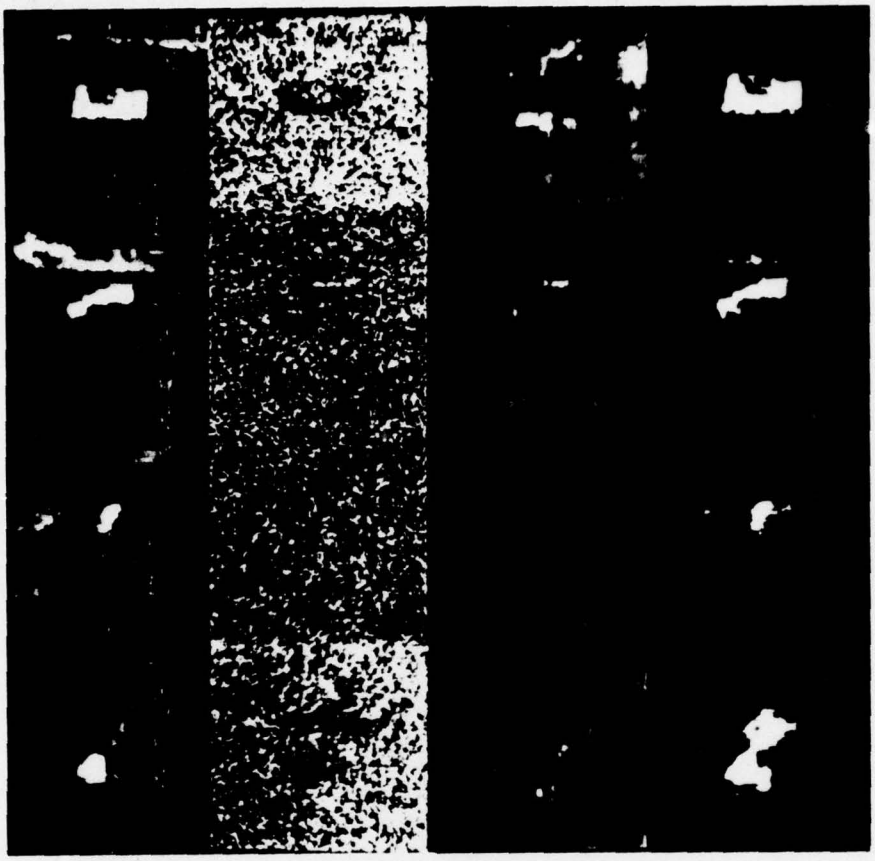


Fig. 8 Composite picture of sixteen 128x128 results using original FLIR images in Fig. 2: (column 1) edge-controlled grey level average over 3x3 window; (column 2) texture extrema; (column 3) texture value picture -- edge-controlled extrema average over 15x15 window; (column 4) segmented results

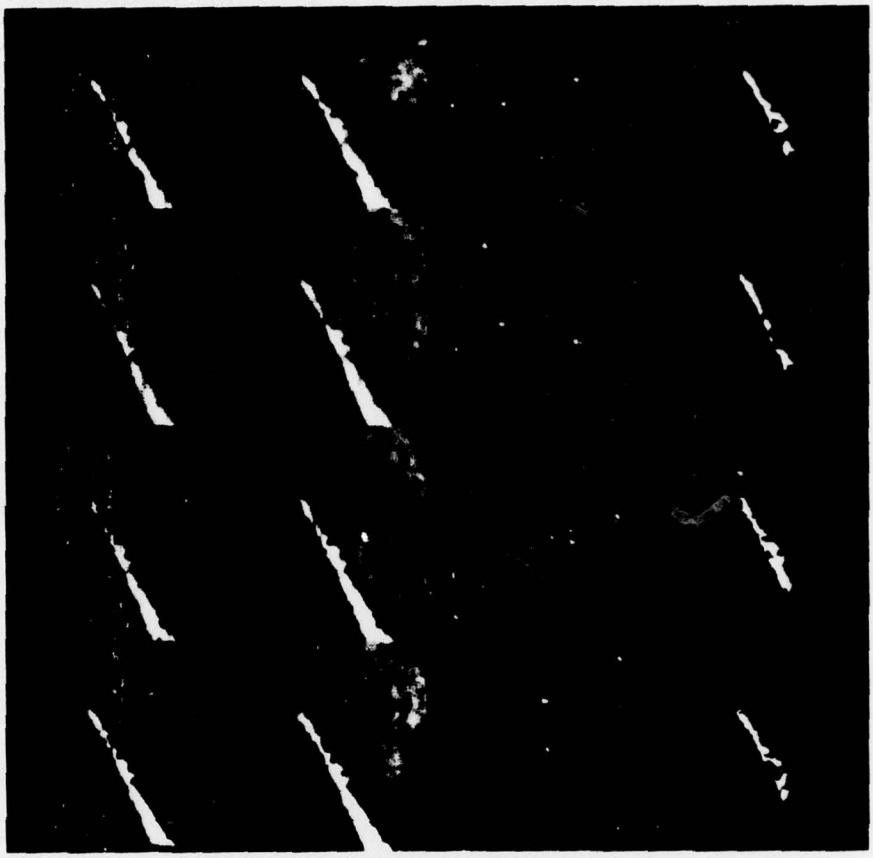


Fig. 9 Sequential frame segmentation of missile: (column 1) 128x128 originals; (column 2) 3x3 edge-controlled average of originals; (column 3) texture value picture -- edge-controlled extrema average over 15x15 window; (column 4) segmented results

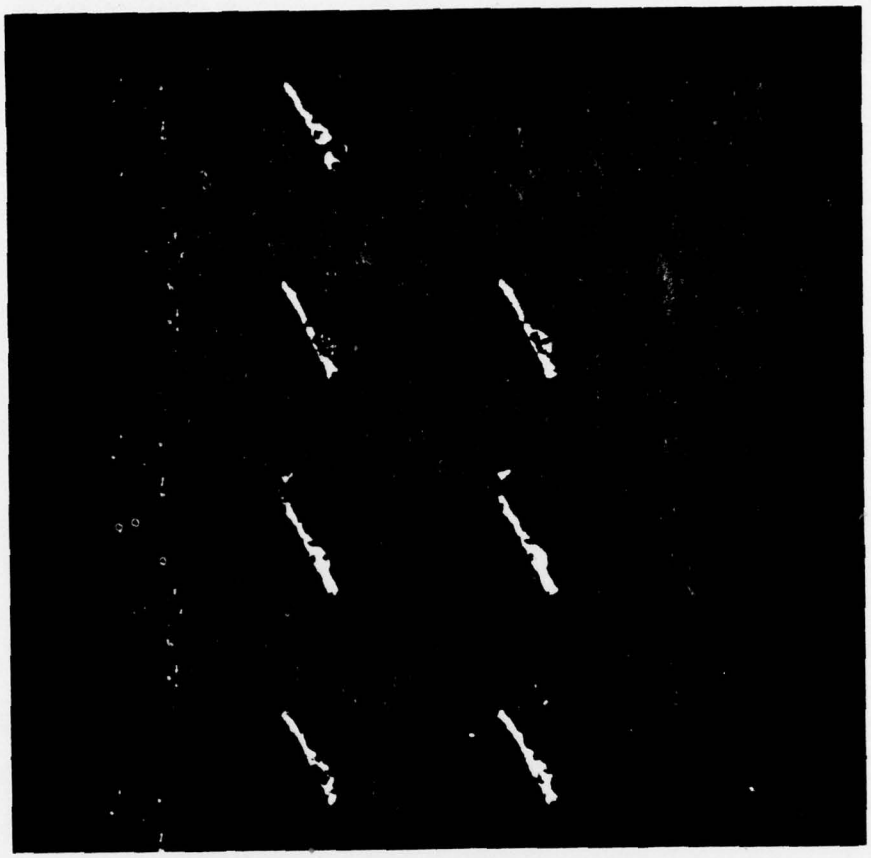


Fig. 10 Sequential frame segmentation comparison of the missile in Fig. 9: (column 1) without prior frame information; (column 2) with prior frame information