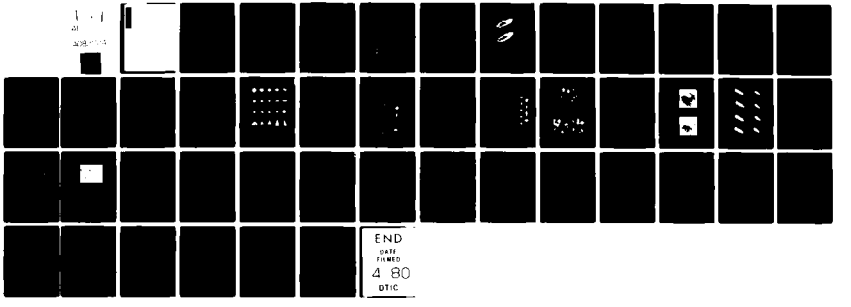
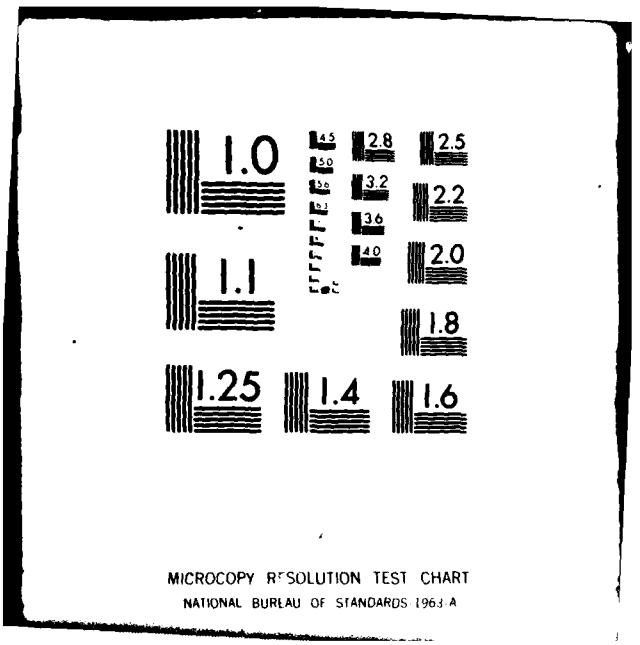
AD-A082 024 NAVAL SURFACE WEAPONS CENTER DAHLGREN VA F/G 17/7
AUTOMATIC TARGET DESIGNATION/RECOGNITION,(U)
OCT 79 R D HILTON
UNCLASSIFIED NSWC/TR-79-272 NL



END
DATE FILMED
4 80
DTIC


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
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963-A

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER NSWC/TR-79-272	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) AUTOMATIC TARGET DESIGNATION/RECOGNITION		5. TYPE OF REPORT & PERIOD COVERED
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) R. D. Hilton		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Surface Weapons Center (K61) Dahlgren, Virginia 22448		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS Computer Support
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Surface Weapons Center (K61) Dahlgren, Virginia 22448		12. REPORT DATE October 1979 13. NUMBER OF PAGES 49
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED 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)

18. SUPPLEMENTARY NOTES

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

optical pattern recognition
target-recognizing weapon
pattern-recognition algorithms

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

The use of pattern recognition techniques for automatic target designation in television-guided weapons is investigated. Computer simulation was used. Target signatures were either drawings digitized by hand or video-taped images of models digitized by machine. A feature-extraction algorithm was designed and used. Encouraging results were obtained from lab equipment, which consisted of a TV camera, a digital-image memory processor, and a minicomputer.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-LF-014-6601

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

LB

FOREWORD

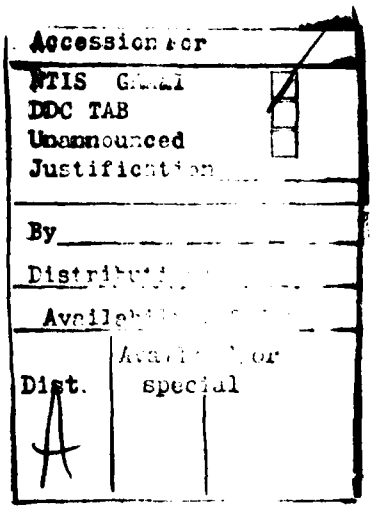
This FY 79 project was designed to illustrate the applicability of pattern recognition technology to fire-and-forget guided projectiles.

Acknowledgment is given to Richard Ely, Electronics Systems Department, for writing the two Basic language programs given in Appendix C.

Reviewers were R. T. Olsen, Acting Head, Computer Engineering Branch, and C. T. Shelton, Acting Head, Computer Facilities Division.

Released by:

R. T. RYLAND, JR., Head
Strategic Systems Department



CONTENTS

INTRODUCTION

The objective of this study is to investigate the possible use of pattern recognition techniques for automatic target designation in television-guided weapons. Pattern recognition is a broad field covering many areas only one of which is optical pattern recognition. One attractive method from this broad field was selected for study.

The method used in the study was computer simulation. Target signatures were either drawings digitized by hand or videotaped images of models digitized by machine.

Pattern recognition is usually thought of in terms of two steps. The first, feature extraction, is the process of making some kind of measurement resulting in definite parameters to be used in the second step. The second step is classification. Here a person designing a pattern recognition system must choose how decisions based upon the selected features are to be made and how many classes can be established for dividing up the patterns received.

The basic framework of the problem treated here is shown in Figure 1. It is assumed that a practical target-recognizing weapon would take the form of a CCD* television sensor and some kind of microprocessor, probably one with microprogrammed instructions. Television sensors of any kind are notorious for producing wide-band output. Pattern recognition algorithms must be very carefully configured in order to avoid having the processing system swamped by the high data rate emanating from the camera.

When a person undertakes to recognize a visual pattern, he tends to overlook some very important subtleties that are automatic in human perception. If one perceives a pattern, such as a rectangular window, it is generally not from a similar rectangle projected on the retina. The position of a person's head (perhaps titled), the orientation of the window (perhaps oblique), and the distance from the eye all combine to transform the perceived pattern into something other than the original when projected onto the retina. A person does not notice these transformations, however.

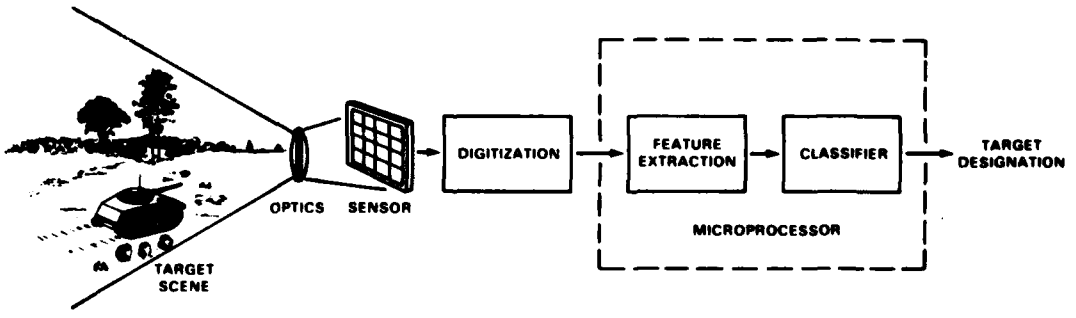


Figure 1. Pattern Recognition Applied to Automatic Target Designation

*Charge-coupled device.

Our perceptive processing, whatever form it may have, is invariant with respect to these transformations. If a machine is to perform optical pattern recognition, it too must be configured to be invariant to such transformations as the optical situation may impose.

Fortunately, methods for deriving mathematical transformations having properties similar to those found in human perception have been published by Hu.[1,2] These transformations, called moment invariants, will be used in this investigation. One transformation that can be incorporated into the invariances is that of vehicle roll, inviting the possibility of a weapon that does not need an elaborate roll-control subsystem.

Another problem area to be addressed is that of treating an optical scene that contains many objects, each of which must be evaluated as a potential target. It is desirable that the separation of the objects be done during a raster scan of the scene rather than by a process requiring random access to the picture elements (pixels) of the scene under evaluation. While random-access television sensors exist, their state of development and economy lag those using conventional TV raster scans. Raster-scanning feature-extraction algorithms are rare in the literature. Work in the area has been undertaken by Scott and Preston[3] and by Agrawala and Kulkarni.[4] A feature-extraction algorithm inspired by but differing from their publications has been designed and used in this study.

MOVEMENT INVARIANT THEORY

The idea of using moment vectors transformed by invariant operators as features for pattern recognition was first suggested by Hu.[1] In this method, the features extracted from a pattern to be classified are moments. Usually the patterns are in the form of silhouettes. A set of moments $\{m_{pq}\}$ are required and, while these are defined by the Riemann integral

$$m_{pq} = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} x^p y^q \rho(x, y)\, dx\, dy \tag{1}$$

(where x,y = cartesian coordinates, and $\rho(x,y)$ = pattern density function), the quantized digital nature of the pattern coordinates and the binary nature of the density of a silhouette mean that the moments $\{m_{pq}\}$ are really computed by

$$m_{pq} = \sum_{i=1}^{N} x_i^p y_i^q \tag{2}$$

(where x_i, y_i = coordinates of ith pixel in pattern, N = number of connected pixels in the pattern).

Summation (2) represents the major compromise taken to reduce the bandwidth of the television signal. Figure 2 shows a potential target that is first quantized in space by the camera's pixel grid and then quantized into a silhouette through selection of a gray level for which a darker shade is made black and a lighter shade is made white. The pattern that enters the pattern recognition process is really a rectangular grid with each site containing a Dirac delta of value 0 or 1. Thus is made the transition from Equation (1) to (2) with all the errors, noise, and ambiguity such processing implies.

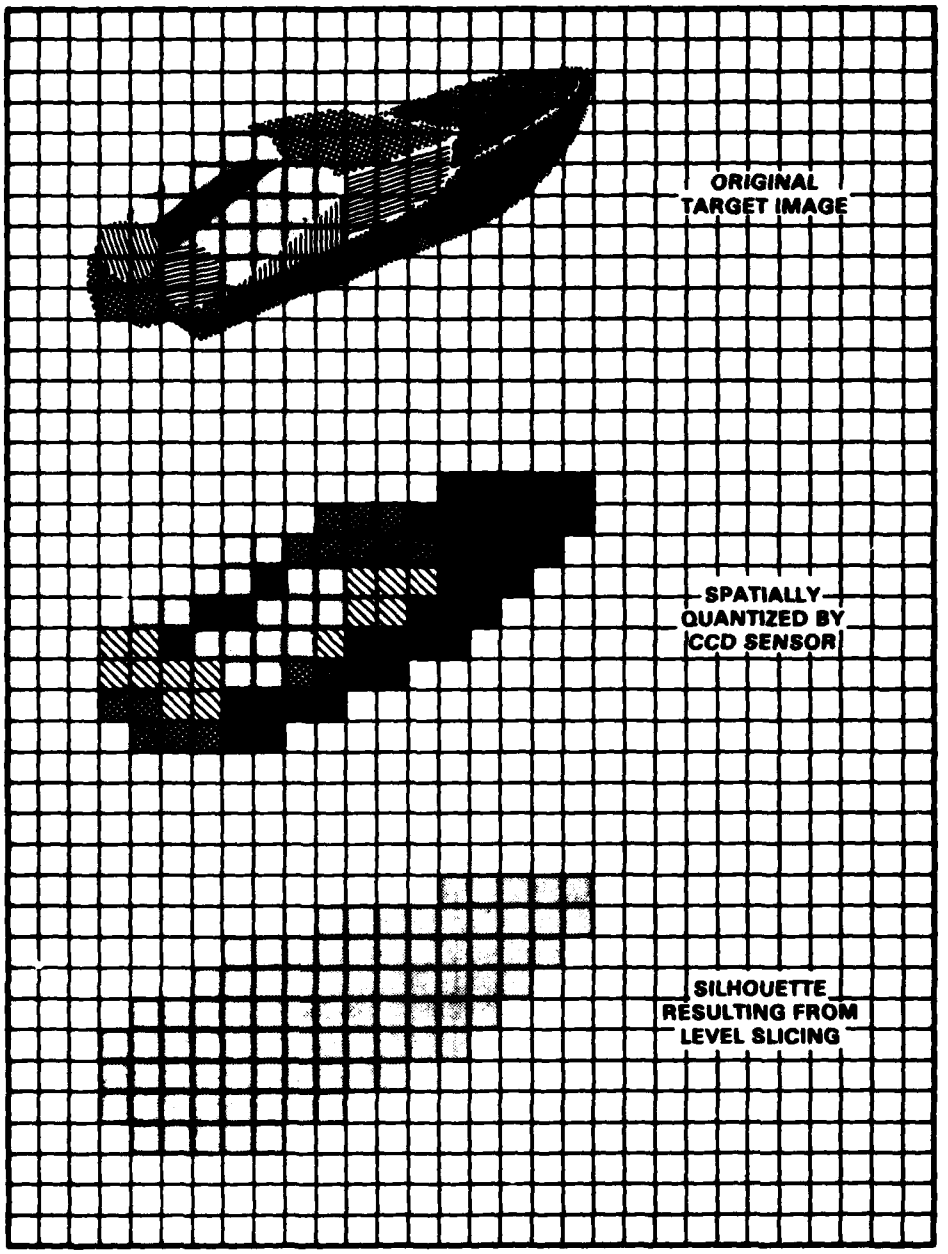


Figure 2. Spatial and Gray Level Quantization of a Target Image

Hu[2] describes the transform to change the set $\{m_{pq}\}$ of ordinary moments into the feature vector $\{M_i\}$ containing moment invariants as an afine transform that can be decomposed into a sequence of elementary transformations. The user of Hu's work may select those elementary transformations that suit his purpose and concatenate them at will within certain restrictions. Elementary transformations from which one may choose include those listed below. The result of each transformation except the last is called a "canonical moment," meaning that the results are moment invariants (functions of moments that are invariant to some parameter such as size) that are themselves moments. Canonical moments can be subjected to still another transformation if needed. Elementary transformations derived in general form by Hu include:

1. Position* (change ordinary moments $\{m_{pq}\}$ to central moments $\{\mu_{pq}\}$
2. Horizontal shear*
3. Vertical shear*

   } (Nullify the effects of perspective)
4. Anisotropic amplification* (Nullify the effect of an oblique view)
5. Size* (Become invariant to distance to object)
6. Rotation (Allow observer to rotate around the optic axis)

Additionally, the user may select the order of the moments ($p + q$ = order) and the size of the set $\{m_{pq}\}$ that are to be employed for feature extraction. The theory is written in terms that are quite general; however, it appears that a practical computational limit may exist with the orders $(p + q)$ reaching three, resulting in a set $\{M_i\}$ of seven elements.

The next section treats the method used to "parse" the picture into its several patterns to be classified. Thereafter we will return to the exact order and size moment invariant set that was derived and evaluated experimentally.

THE SHORT-LIST ALGORITHM

The short-list algorithm is a process designed to extract from raw video data the necessary sets of moments $\{m_{pq}\}$ for each recognizable pattern within the field of view. The main requirement for such an algorithm is that it be as simple as possible in order to be able to process a frame of video data in the shortest possible period of time. In a weapon, this algorithm would probably be incorporated into a single microprogrammed microprocessor instruction.

The video data, as it reaches the short-list algorithm, is assumed to have been sliced at some appropriate gray level so as to contain a background of "0" pixels with "1" pixels forming patterns. Figure 3 is a computer printout that illustrates what a portion of such a frame might look like. The background pixels are blank and the "1" pixels are shown as asterisks. Connected groups of "1" pixels are considered to comprise patterns to be processed for recognition. Pixels are considered connected if they join similar pixels above, below, or to either side. Figure 4a contains five pixels that are connected, and Figure 4b contains five pixels that are not.

*Canonical.

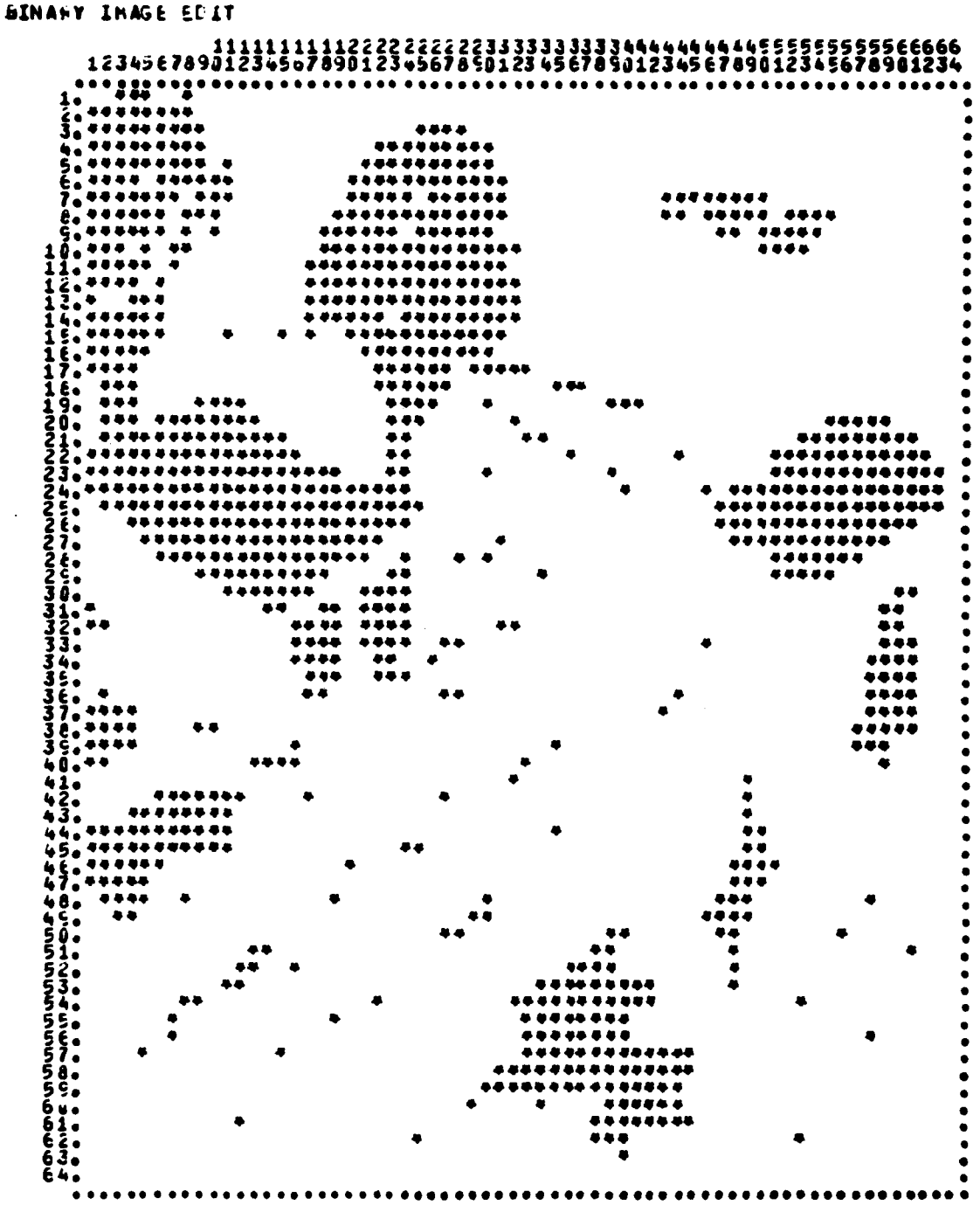


Figure 3. Edit of a Typical Binary Image

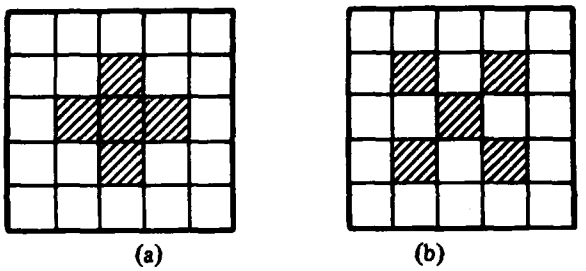


Figure 4. Connected and Unconnected Pixels

The algorithm receives a train of 1 or 0 pixels in sequence as the raster scan proceeds. It has been found that a zig-zag scan such as that shown in Figure 5 is best for this purpose. The algorithm maintains a table that at the end of the scan will contain sets of moments for each pattern found. Each row or entry of the table contains three control quantities U, R, and L plus the required number of moments $\{m_{pq}\}$. Initially, the contents of the table are all set to zero. As pixels arrive, each "1" pixel is assigned an entry in the table and the moment values $\{m_{pq}\}$ are incremented with the appropriate quantity according to Equation (2). The algorithm strives to assign connected pixels to the same entry of the table and to consolidate entries that are found to be parts of a common pattern.

For purposes of checkout and early conceptual design, the algorithm has been programmed in Fortran and provided with printouts that monitor its operation. As the image is scanned and "1" pixels appear, each "1" pixel is assigned an entry in the moment table and the summations performed. Each entry in the table is assigned a character as address identification, and a printout of the binary image inserts the assigned character into the corresponding place in the image.

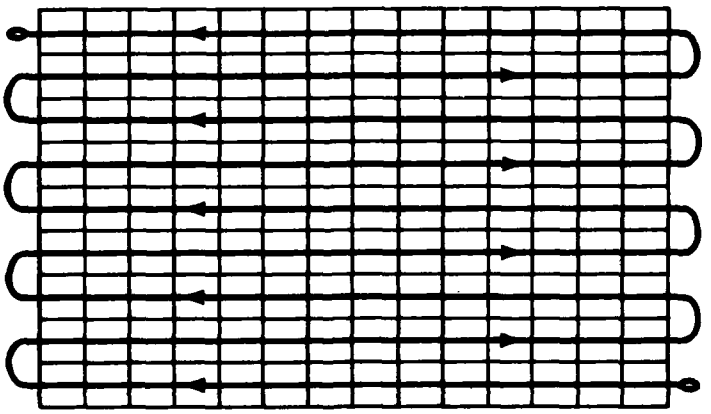


Figure 5. Zig-Zag Raster Scan Used in the Short-List Algorithm

Character assignments are made with knowledge of only two pieces of information: (1) the character assigned to the previous pixel scanned and (2) the character assigned to the pixel just above the current one. When a "0" pixel is received, no summation takes place and the printout always shows a blank character assignment. When a "1" pixel is received, the following rules prevail.

1. If the previous character P and the above character A are both blank, assign a new character. To assign a new character, search the moment table for the lowest entry not in use by seeking an entry with the U code equal to 0. In making the character assignment, set U = 1 and do the summations.

2. If either A or P are nonblank and the other is blank, assign the nonblank character and do the summations.

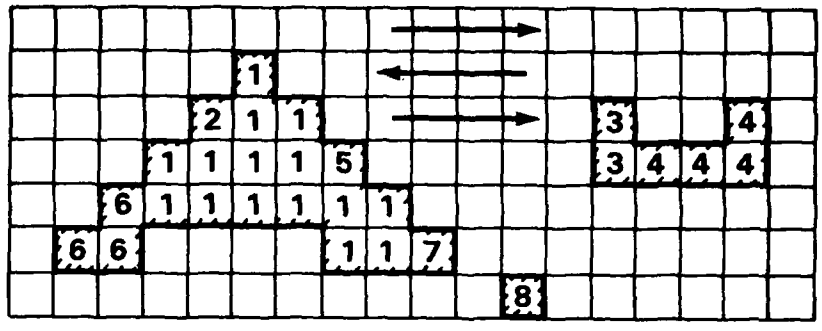
3. If A and P are nonblank and equal to each other, assign that equal character and do the summations.

4. If A and P are both nonblank but not equal to each other, assign the smaller of the two and do the summations. Also, increase the U code in the assigned entry by 1 and enter that assigned character into the R or reference column of the greater of P or A.

Figure 6 shows a sample image that contains two patterns of connected "1" pixels and an isolated "1" pixel. Also shown is the moment table resulting from a scan and the above rules. Only m_{pq} with $(p,q) = (0,0)$ is shown among the moments. This 0th moment is equal to the total number of pixels assigned that character. If one now scans the entries from the higher characters to the lower and, upon finding an entry with reference $R \neq 0$, merges the moments of that entry with those of the entry referenced and further clears that entry with that reference to all zeros, then the result will be a table containing one active entry ($U \neq 0$) for each pattern in the image.

The algorithm as just described does deliver a table containing the necessary moments $\{m_{pq}\}$ for pattern recognition and target designation. It is not economical with memory space, however. Isolated "1" pixels like entry 8 and once-used entries like 2, 5, and 7 remain active through the process using up valuable moment table space. To make better use of memory resources, a process called purging was introduced. It also turns out that purging introduces the possibility of a malfunction. It has been found that images that contain, among many other patterns, objects that are long, twisted, and narrow may be incorrectly processed in that the long patterns are occasionally cleaved at narrow, one-character-wide points. Malfunctions have not been observed within the more compact patterns that are likely to be identifiable targets, however, and the purging process has been kept.

Purging is a process that is invoked periodically during a raster scan. The period that has been used in the Fortran simulation is one per scan line. Longer periods may be optimal, but no exact determination has been made. The object of the purge is to remove from the table those entries that are not in current use (i.e., not used since the last purge) and either represent objects too small to be considered for recognition processing or entries that can be combined with other larger entries. Entries that are not in current use are identified by means of the flag L in the moment table. Each time an entry is used for a moment summation, that flag is set to 1. As each active entry is addressed during a purge, L is set to 0. Thus, when an active entry ($U \neq 0$) is addressed



MOMENT TABLE

Ch	U	R	L	m_{oo}	m_{ij} ....
1	5		—	15	
2	1	1	—	1	
3	2		—	2	
4	1	3	—	4	
5	1	1	—	1	
6	1	1	—	3	
7	1	1	—	1	
8	1		—	1	
9	0		—		
A	0		—		
.....					

Figure 6. Sample Image with Character Assignment and Moment Table Entries

that has L = 0, it is known that this entry has not been used since the previous purge and may be considered for elimination.

The printout from the Fortran simulation identifies those entries purged at the end of each scan line by printing their identifying characters in a row to the right of the image frame. The progress of the dynamic allocation of memory space can thus be observed.

The logic of the purge and merge process is as follows:

1. Address each entry in the moment table in descending order.

2. If $U \neq 1$, ignore entry. If U = 0, the entry is not active; if $U > 1$, then it is referenced by other entries and must be kept.

3. If U = 1 and L = 1, set L = 0 and go on to the next entry.

4. If U = 1 and L = 0 and R ≠ 0, combine the moments of this entry with those in the referenced entry and set all the contents of this entry to zero (purge).

5. If U = 1 and L = 0 and R = 0 and $m_{00} < S$, set all the contents of this entry to zero. S is that number of pixels needed for significant size in the recognition process.

6. If U = 1, and L = 0, and R = 0, and $m_{00} \geq S$, go on to next entry.

Figure 7 shows the printout from a Fortran simulation using the above rules on the data shown in Figure 3. The useful result is, of course, the contents of the moment table at the end of the scan. That table appears in Figure 8. It can be seen that the copious amount of information from the television sensor has been reduced to a table of numbers, the data format in which the microprocessor performs best. The short-list algorithm is thus a preprocessing step used in order that the final selection process can function without being swamped with details. Similar preprocessing is really employed by people in most decisionmaking situations. For this reason, the short-list algorithm was named after the title of Chapter 5 in the famous work on decisionmaking by C. Northcote Parkinson.[5]

N-ORDER MOMENT HARDWARE

If a process like the short-list algorithm is to be implemented in conjunction with an image sensor, then some consideration of special hardware becomes necessary. A very slow imaging device such as a millimeter wave sensor would issue video data at a slow enough rate to permit the algorithm to be implemented with ordinary programming techniques. A CCD sensor mounted aboard a missile would be in a different situation, however. A possible implementation might employ a bit slice microprocessor with microprogrammed instructions. If a 256 X 256 sensor is used, then values of x^p, y^q for $0 \leq x \leq 255$, $0 \leq y \leq 255$ would be repeatedly needed. While y^q need be recomputed only for each row, x^p values repeat over and over again. Recomputation of x^p values might well be avoided by using a hardware multiplier or a read-only memory. Obviously, quantities like $(255)^3$ require rather large word sizes (24 bits) and the associated hardware has corresponding expense. Also, third-order moments imply a set $\{m_{pq}\}$ of 10 elements, four of which must accept 24-bit increments. On the other hand, very fine discrimination among various types of aircraft using third-order moments have been reported.[2,6]

SECOND-ORDER EXPERIMENTS

Since the implementation of third-order moment computation involves a significant increase in system complexity over what would be required for only second-order moments, a second-order moment invariant set was implemented and experimented with as a preliminary step.

Starting with an ordinary moment array from the short-list algorithm, the following invariants were computed in the Fortran simulation program.

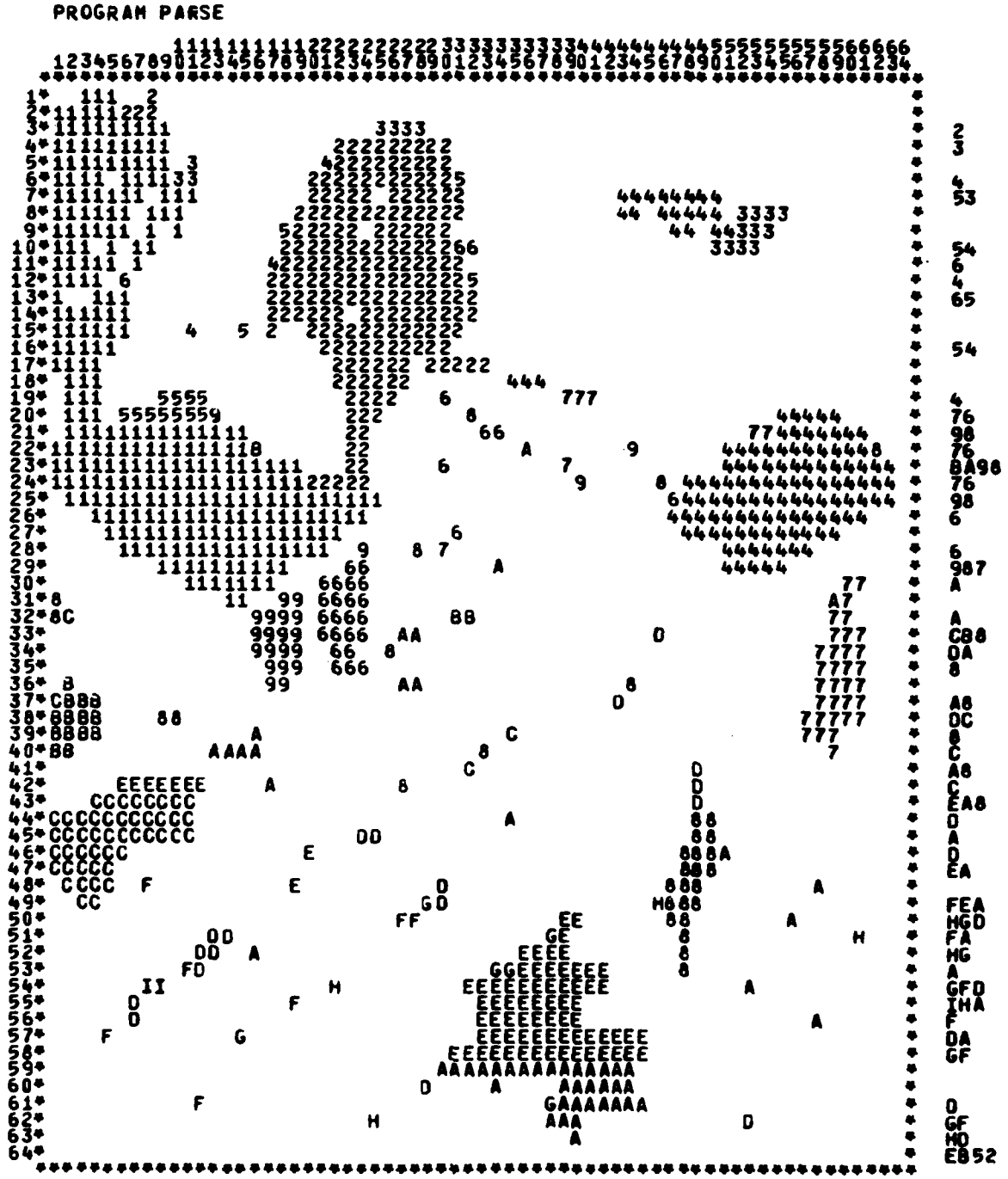


Figure 7. Image Printout from the Short-List Algorithm

LOG	CH	U	MEMORY REF	L	M00	M01	M02	M03	M10	M11	M12	M20	M21	M30
1	1	10	0	0	512	7977	158083	3537087	7085	117623	2229521	164467	2311909	3928927
3	[illegible]	3	0	0	30	247	2863	17679	1474	12187	182207	72766	604059	3600782
4	4	5	0	0	111	2716	67076	1671950	6135	149782	3691032	368887	8304424	19960049
6	6	1	0	0	24	763	24340	779947	545	17320	552548	12483	394610	202869
7	7	2	0	0	34	1280	42684	1521198	2019	71219	2827117	119939	4220467	7127719
8	8	1	0	0	26	1225	57937	2750455	1263	59462	2010174	61385	2087862	2900033
9	9	1	0	0	19	635	21265	713555	336	11227	375097	9962	190163	180140
10	A	15	0	0	106	6022	343044	19593412	4843	229903	13109237	159943	8854129	6034169
12	C	6	0	0	54	2418	108470	4874060	312	13786	610132	2324	181760	19872

Figure 8. Moment Table from the Short-List Algorithm

Given $\{m_{pq}\} = m_{00}, m_{01}, m_{02}, m_{10}, m_{11}, m_{20}$, convert the second-order ordinary moments to central moments μ_{pq}. By definition,

$$\mu_{pq} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} (x - \bar{x})^p (y - \bar{y})^q f(x, y)\, dx\, dy \qquad (3)$$

(where $\bar{x} = x$ centroid $= m_{10}/m_{00}$, and $\bar{y} = y$ centroid $= m_{01}/m_{00}$).

By integrating Equation (3) by parts, it can be shown that

$$\begin{aligned} \mu_{02} &= m_{02} - \bar{y}^2 \\ \mu_{11} &= m_{11} - \bar{x}\bar{y} \\ \mu_{20} &= m_{20} - \bar{x}^2 \end{aligned} \qquad (4)$$

These results are moment invariants (invariant with respect to position) which are moments themselves (canonical moments), so we are justified in applying a second transformation to them. This second transformation is one of size invariance, and for general (p,q) it takes the form

$$\mu'_{pq} = \frac{\mu_{pq}}{m_{00}^{\left(\frac{p+q}{2}+1\right)}}, \quad p + q > 2\,. \qquad (5)$$

For our second-order moment invariant set

$$\mu'_{02} = \frac{\mu_{02}}{m_{00}^2}$$

$$\mu'_{11} = \frac{\mu_{11}}{m_{00}^2}$$

$$\mu'_{20} = \frac{\mu_{20}}{m_{00}^2}$$

Finally, we desire rotation invariance. This final transformation is not canonical because the products of the process are not moments but merely functions of moments. We must be content to have this be the final transformation. For second-order, there are only two moment invariants. They are

$$M_1 = \mu'_{20} + \mu'_{02}$$

$$M_2 = (\mu'_{20} - \mu'_{02})^2 + 4\mu'^2_{11} .$$

A proof of the validity of these final invariances appears in Appendix A.

The two variables (M_1, M_2) will now be used as features extracted from patterns in the original picture and made invariant to three effects of the variable viewpoint from which they might have been observed.

Various patterns were generated by hand and submitted to a Fortran program which entailed (1) a punch card input format of one card per scan line of 64 binary pixels, (2) the short-list algorithm, and (3) Equations (5), (6), (7). Details of the program appear in Appendix B.

Each pattern generates a point in two-dimensional (M_1, M_2) feature space. Experimental steps were undertaken to see if patterns that might resemble targets tend to cluster or otherwise distinguish themselves in feature space.

An array of digital approximations to various geometric shapes was prepared and fed into the program. The shapes were compact solid patterns that were thought to be representative of man-made objects likely to be targets. Some of these patterns together with their resulting point in (M_1, M_2) feature space appear in Figure 9. Compact, fairly symmetrical patterns such as these generate points in a rather restricted area of feature space, and the loci of some of them appear in Figure 10. The obviously symmetrical patterns like the circle, the square, and the equilateral triangle all lie on the M_1 axis, since M_2 is a measure of elongation. The solid circle occupies the lowest position on the M_1 axis because this shape has the smallest possible radius of gyration for a given area. A square has a greater radius of gyration for a given area, and an equilateral triangle has a still greater radius of gyration. Symmetric hollow patterns can lie anywhere on the M_1 axis that exceeds the position occupied by the solid circle.

As the basic symmetrical patterns are elongated, their point in (M_1, M_2) feature space follows a locus upward and to the right. The three loci shown in Figure 10 resulted from elongating the circle, the square, and the equilateral triangle. The rectangle composed of a 7 by 12 matrix of pixels was subjected to additional distortions. The point in feature space generated by the undistorted rectangle is the solid dot beside the third rectangle from the bottom. Quantized approximations of this same shape in two different rotated positions appear in Figure 9, and their points in feature space are plotted in Figure 10 as circles and connected to the unrotated point by dotted lines. The 7 by 12 rectangle was also skewed into a rhomboid, and its point in feature space is plotted as a hollow square and connected to the original by a dashed line. Applying skew has a similar effect in

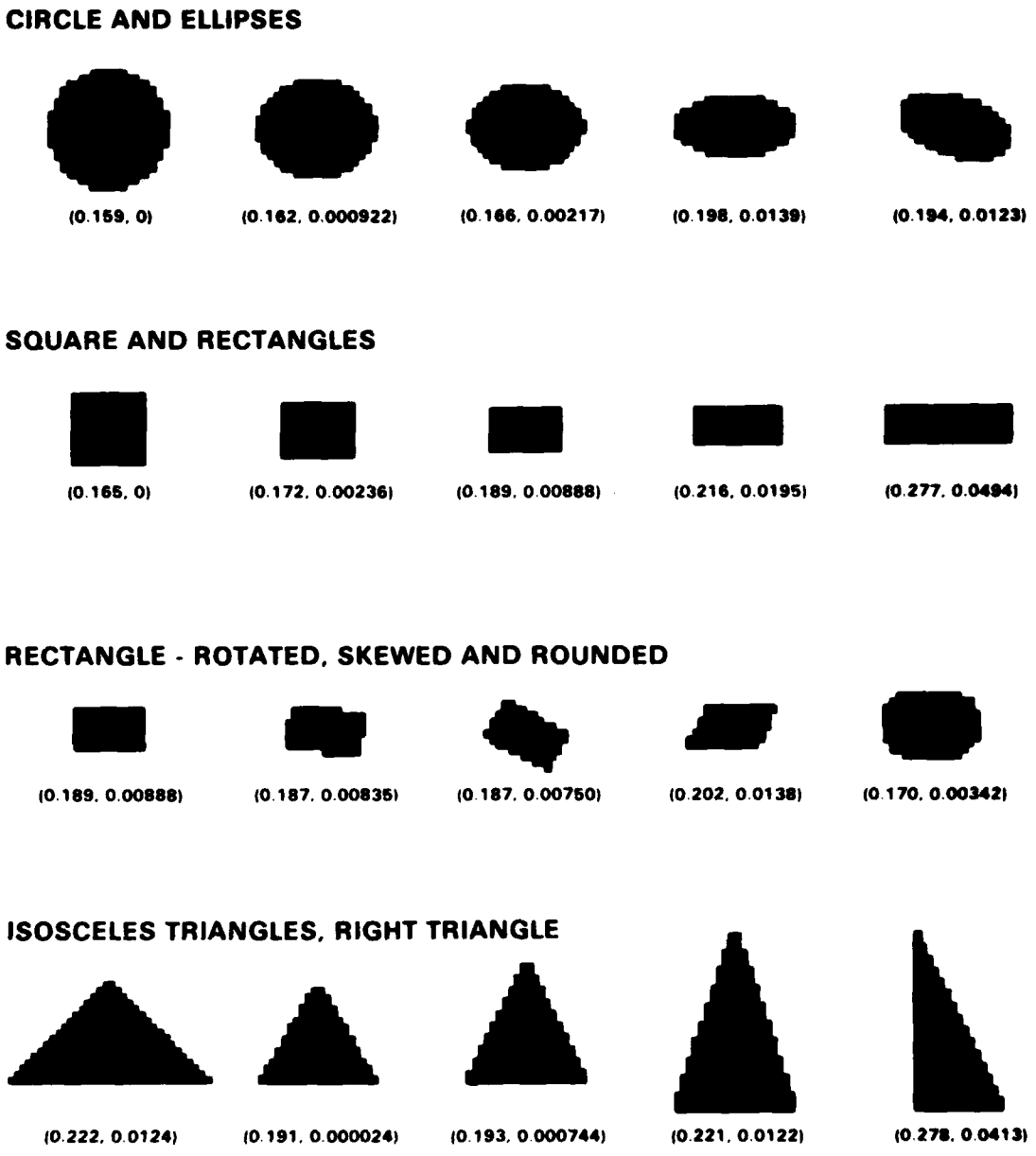


Figure 9. Geometric Patterns and their Second-Order Feature Space Coordinates

feature space as elongation. This represents an important extra invariance since flat objects viewed from an oblique angle have a skewed appearance. A similar effect is noticed in triangles. While the equilateral triangle lies on the M_1 axis, other triangles occur on a locus emanating from that point. As a result of the "free" skew invariance, isoceles triangles and right triangles share the same locus except that a right triangle must lie above that point occupied by a 45° right triangle.

The discrimination power of second-order moment invariants for land targets was considered. Figure 11 shows some sketches of what a tank might look like at low resolution and at several orientations. Also included in the sketch are a few shapes that might show up in a scene containing isolated patches of dark vegetation such as trees against a grass background. These items are shown as silhouettes in Figure 12, which also shows their respective points in feature space. The loci of ellipses, rectangles, and triangles in feature space are also shown for reference. While the tank shapes generally follow the rectangle locus and two of the trees (B and C) are out of the feature space neighborhood shown, the other two trees are within the region that is occupied by tanks. The hatched region between the ellipse and rectangle loci and below $M_2 = 0.018$ is considered to be "characteristic of tanks" and perhaps also characteristic of silhouettes that bear some resemblance to those of tanks.

As an additional indication of the kind of discrimination that might be expected from second-order moment invariants, another drawing was prepared and processed. This drawing, which appears

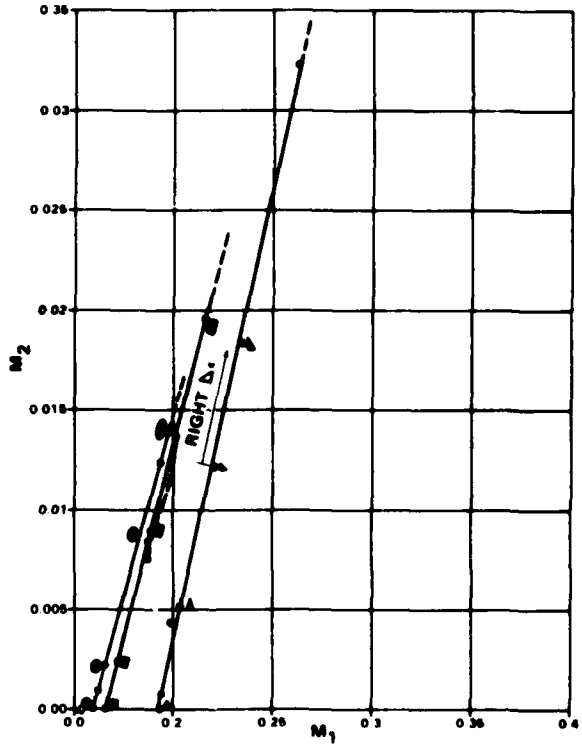


Figure 10. Moment Invariant Loci for Various Solid Geometric Shapes

in Figure 13a, represents a possible section from a land target area. It includes a target shape (a tank) and an array of other items such as barnyard paraphenalia and shrubs. This drawing was hand-digitized in two different orientations (Figures 13b and 13c) representing two roll angles of the picture-taking vehicle. The moment invariants M_1 and M_2 were computed for each of the patterns in these digitization and the results are plotted in Figure 14. The digitization noise caused some points to shift in the two orientations. Points representing like patterns are connected by a dashed line in Figure 14. It can be seen that shapes other than the tank have intruded into the area in (M_1, M_2) space characteristic of tanks. Many other shapes were completely eliminated in this discrimination process as well. Actually there is a third feature that could be used to aid in the discrimination and that is the ordinary moment m_{00}. The 0th moment m_{00} is the pixel count for the pattern, and if altitude is known at the picture-taking time (perhaps from barometric pressure) then patterns having an area inconsistent with the anticipated target size could be eliminated.

The final target scenario considered was at sea. Clutter there is much less in evidence, and the problem of partly occluded targets all but disappears. Patterns, if they are a target, will tend to have

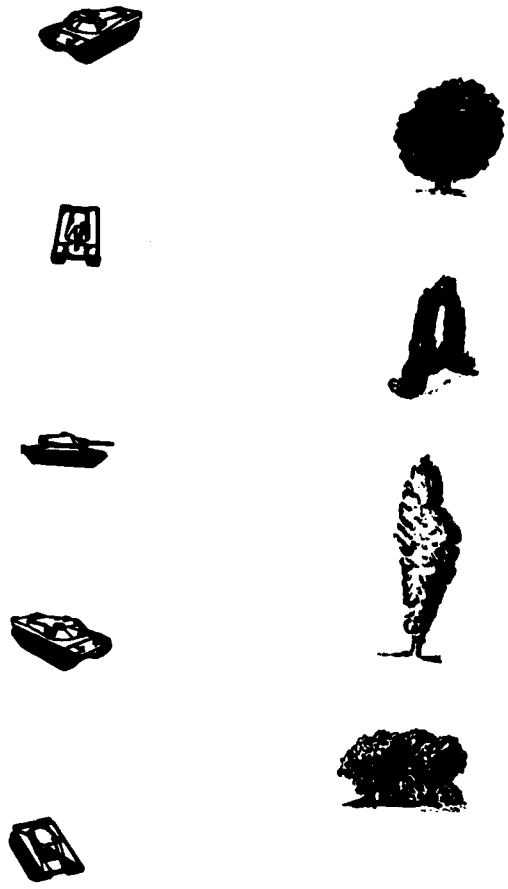


Figure 11. Some Sketches of Possible Land Targets and Possible Clutter

a characteristic boat shape. Other objects, such as small islands, shorelines, rocks, and docks, can be expected to have very different shapes and very different mappings into (M_1, M_2) space.

Psychological testing has been done at the Naval Surface Weapons Center to determine the target-discrimination capacity of a man viewing a sea target through a television monitor. One such test was to discriminate between two models shown in Figures 15 and 16. They roughly represent the eastern KOMAR- and OSA-type ships, respectively. These photos were digitized by hand so as to have four different sizes of target image. The digitized silhouettes appear in Figure 17. The eight patterns are designated K_N and O_N = 1, 2, 3, 4, and the numbers of pixels in each are tabulated below.

N	K_N	O_N
1	364	671
2	229	417
3	54	105
4	36	67

The two sets of points representing the various digitizations in (M_1, M_2) space appear in Figure 18. Clearly, the two types of target are well separated when viewed at the particular aspect in the original photos.

Finally, equipment became available for a test using an electronically digitized television image. Figure 19 is a block diagram of the setup. A model sea scene was made with a piece of photographic backdrop paper as water, a piece of shag carpet as an island, and three small ship models. The ship models available were of YORKTOWN, ENTERPRISE, and MISSOURI. They were about 15 cm in length and shaped so as to rest on a surface at waterline.

The lab equipment consisted of a television camera, a Quantex digital-image memory processor model DS-20, and a Tektronix minicomputer model 4051. The Quantex device is capable of digitizing one frame of video and storing the result in internal memory (6 bits per pixel). The Quantex memory can, in turn, be interrogated by the Tektronix 4051. Two basic language programs, listed in Appendix C, were used to extract and format the data. Figure 20 is an example of the program output. A 128 × 128 pixel area, chosen by the user, is averaged into a 64 × 64 pixel output array, which is presented as a 10 gray level printer plot.

Figure 21 is a photograph of the sea model taken through the television monitor. The elevation angle of the optic axis was about 45°. Three images were digitized. Each contained all three ships and the island. Each picture featured a different orientation of the ships so that end-on views, broadside views, and various other angles were included.

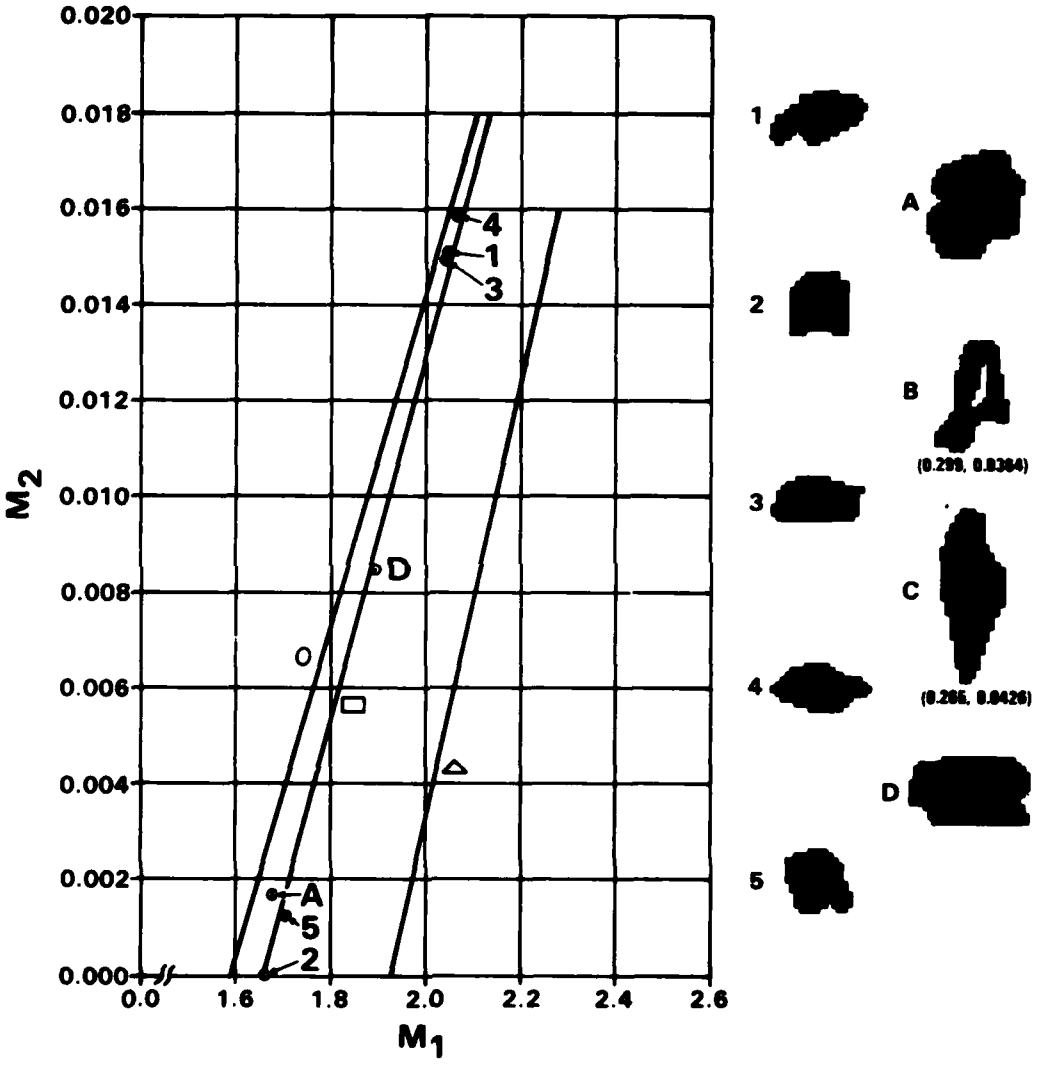
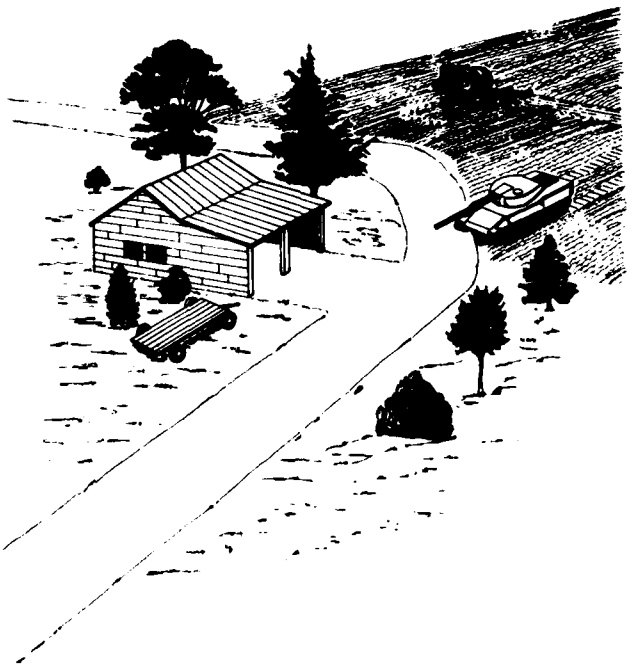
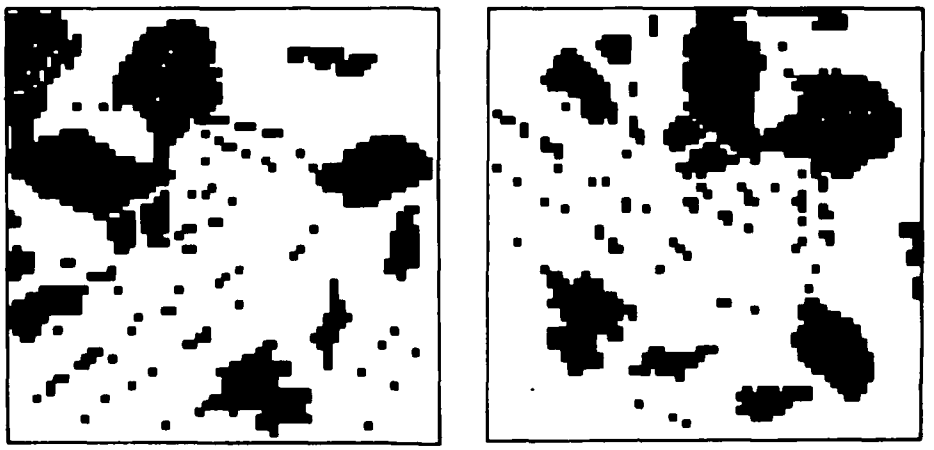


Figure 12. Land Targets as Silhouettes and as Points in Feature Space



(a) Drawing



(b) Hand Digitizations in
Different Orientations

Figure 13. Barnyard Scene

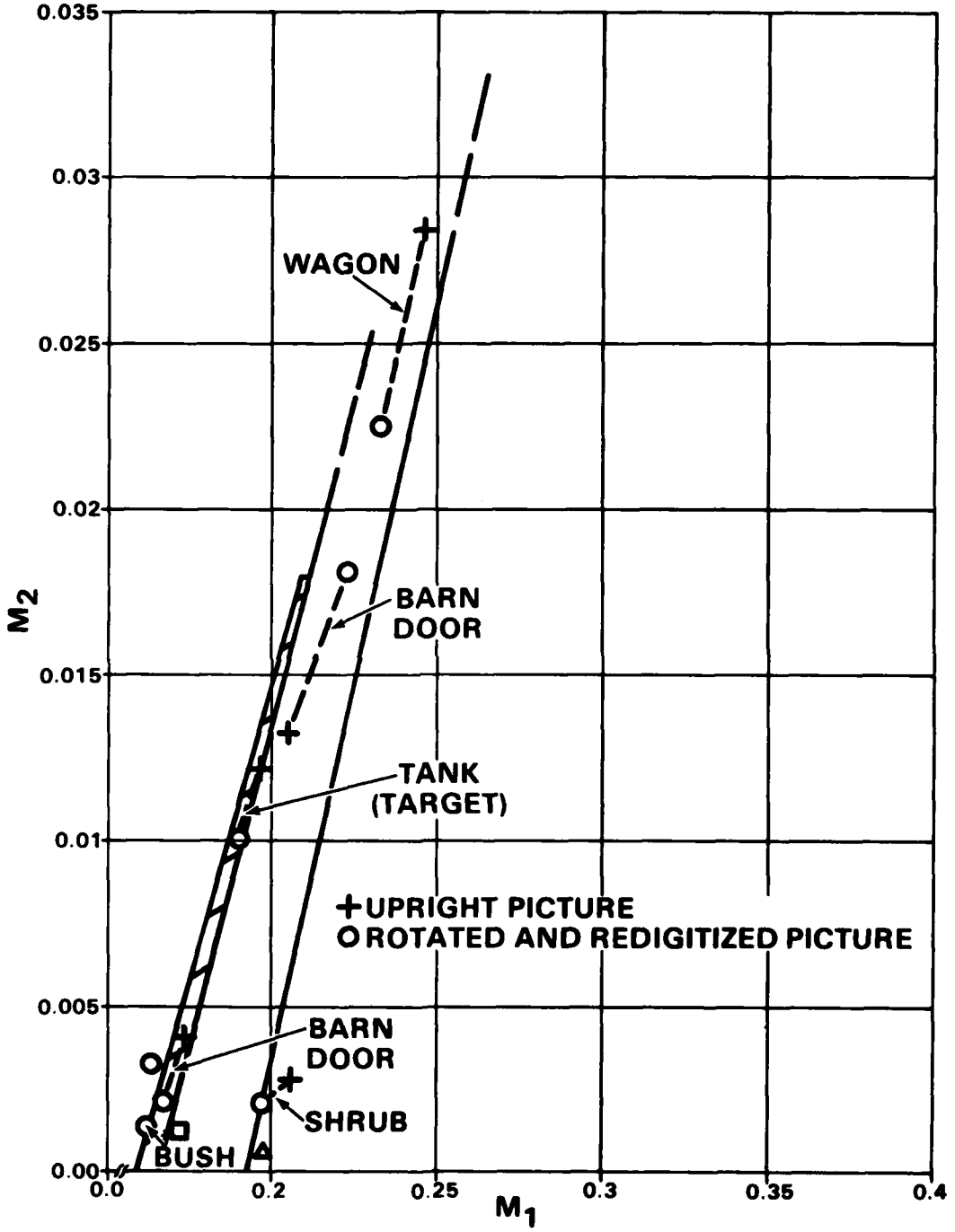


Figure 14. Moment Invariants from Barnyard Picture

Figure 15. KOMAR Model

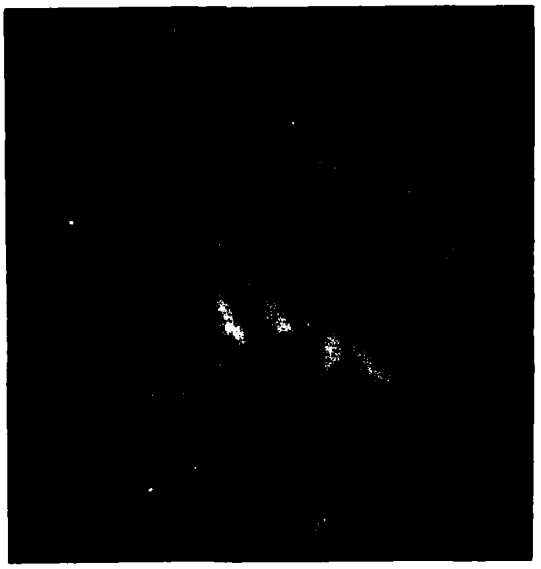


Figure 16. OSA Model

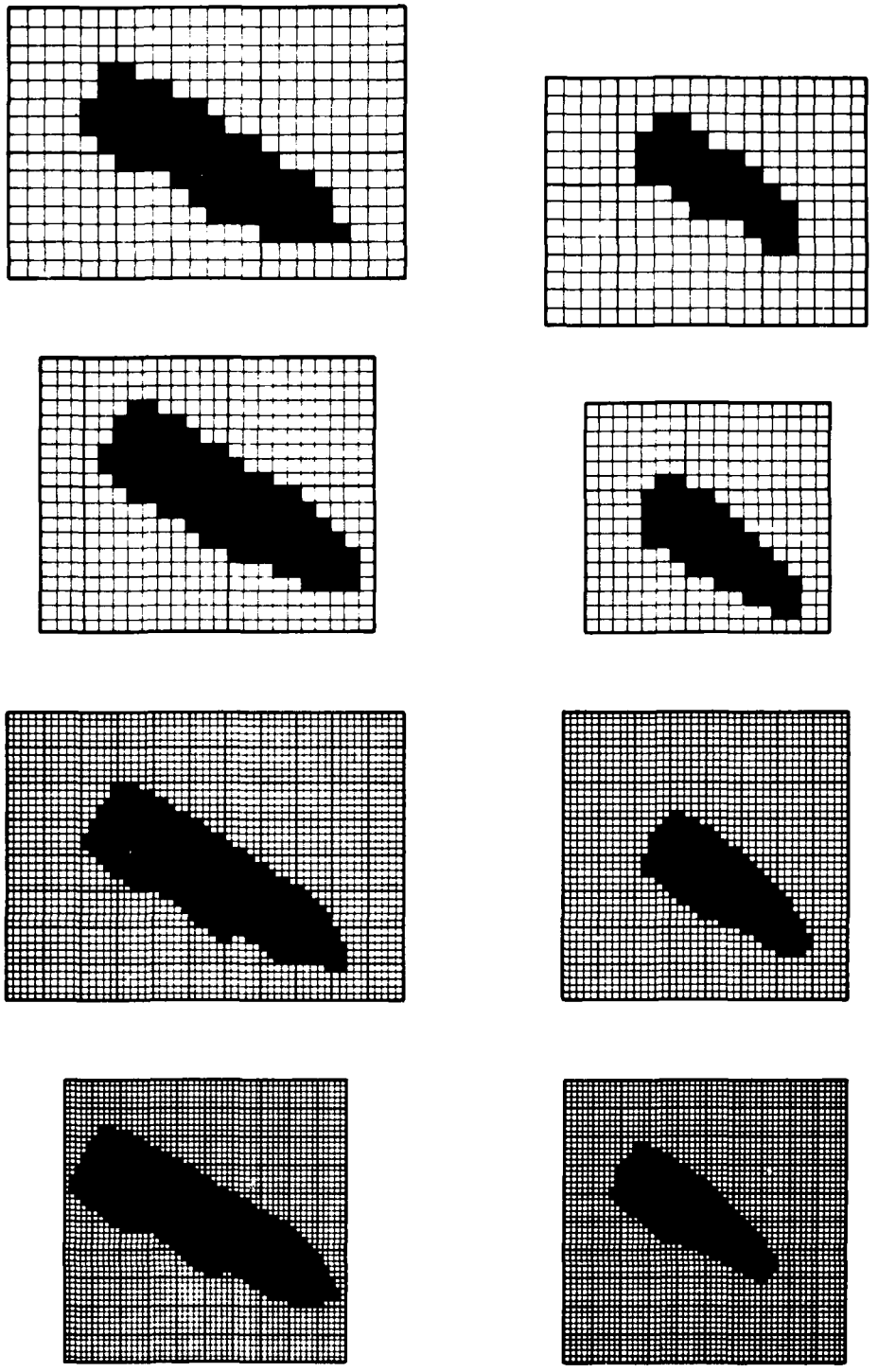


Figure 17. Digitized Silhouettes of KOMAR and OSA

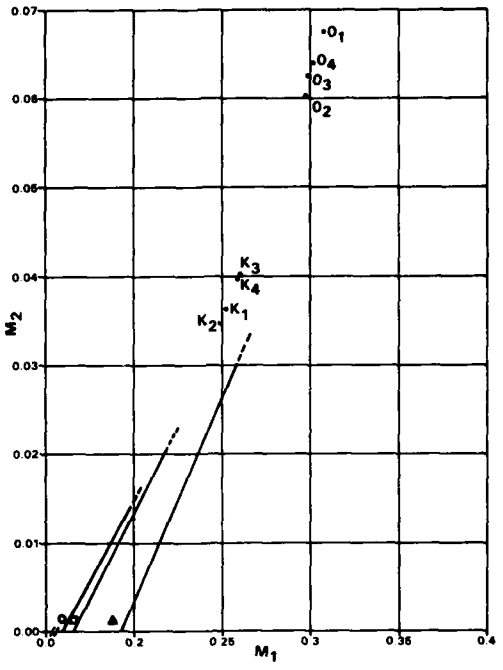


Figure 18. Digitization in Space of KOMAR and OSA

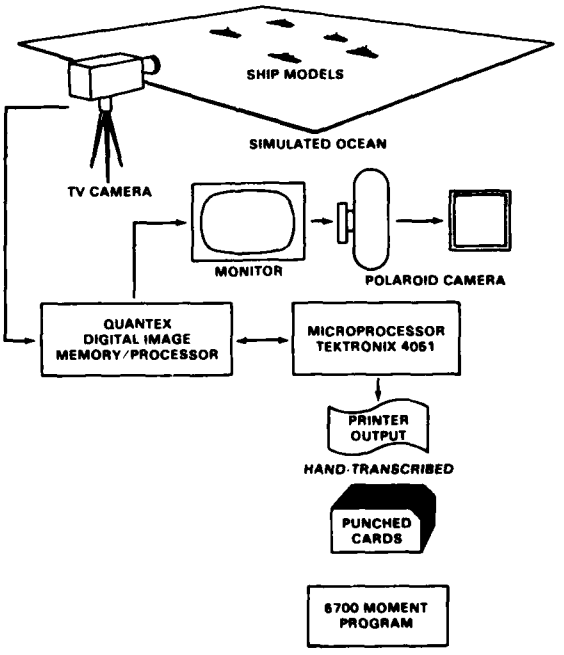


Figure 19. Target Recognition, Digitization Equipment

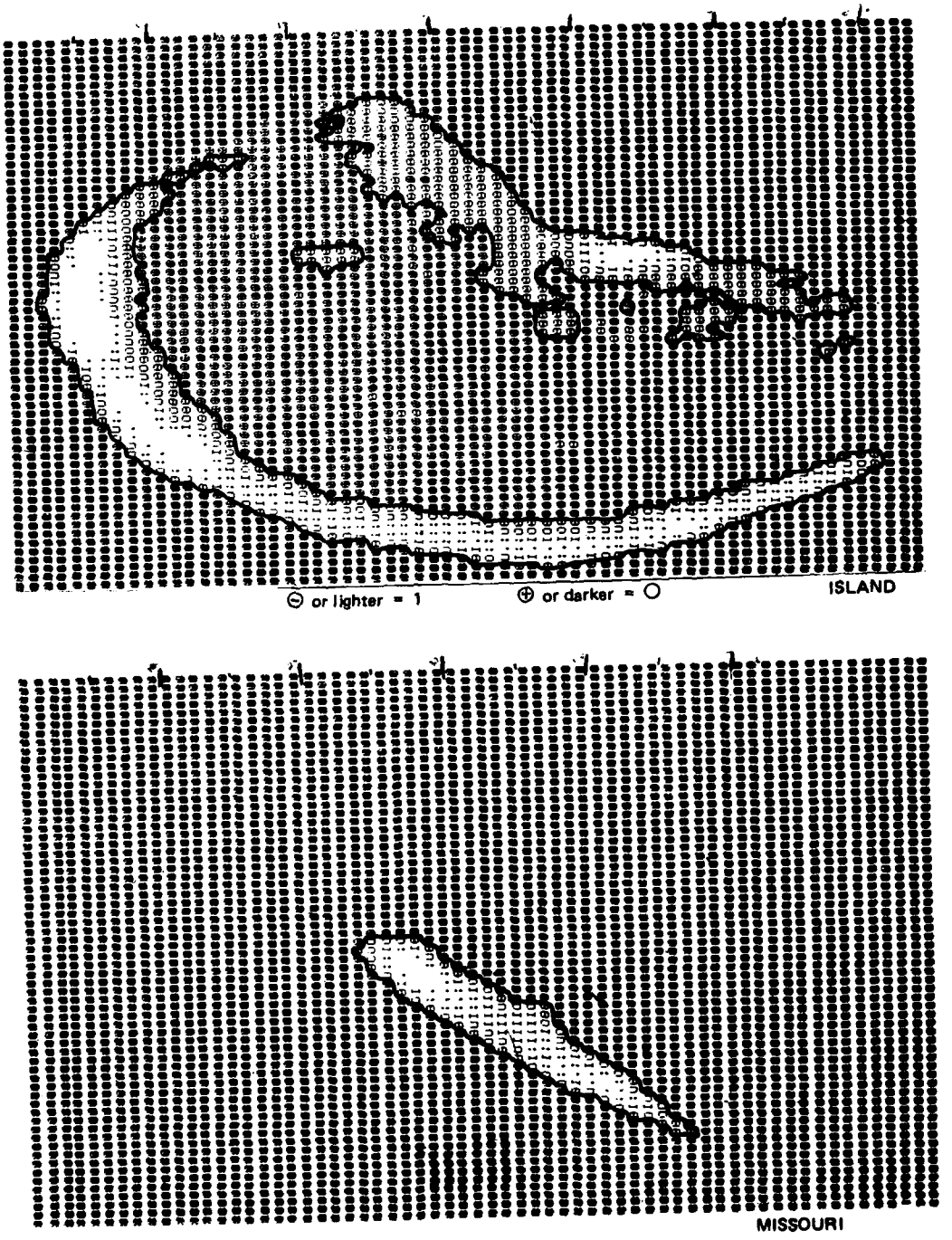


Figure 20. Program Output of Model Sea Scene

Figure 21. TV Image of Sea Model

The resulting printouts from the program were transferred to punched cards for the simulation program used earlier for drawings. Any light level represented by a θ (- overstruck by a 0) was transcribed as a 1. The resulting points in feature space appear in Figure 22. The nine ship mappings are plotted as large dots. For comparison, the feature space points for OSA and KOMAR are also plotted as small dots. The island beaches were seen as two patterns by the program. The feature space mappings for these patterns are plotted as triangles. It can be seen that the boats and ships fall within a narrow corridor in feature space while the beaches lie well outside this region.

CONCLUSIONS

The results from these tests are considered encouraging. A microprocessor can easily interpret the position of a feature space point in terms of target likeness and determine the most likely target among an array of such points. Having done that, the microprocessor has enough data to designate a target to a more or less conventional TV guidance system.

The approach in this investigation has been a heuristic one. Formal pattern-recognition theory embraces other more analytical procedures, and their application to the recognition of naval targets should be investigated. For example, once target shape probabilities and clutter density can be quantitively assessed, there exist definite procedures for optimizing the classification criteria in feature space and for optimizing the number of dimensions of that space. An appreciable payoff for going to third-order moments may exist.

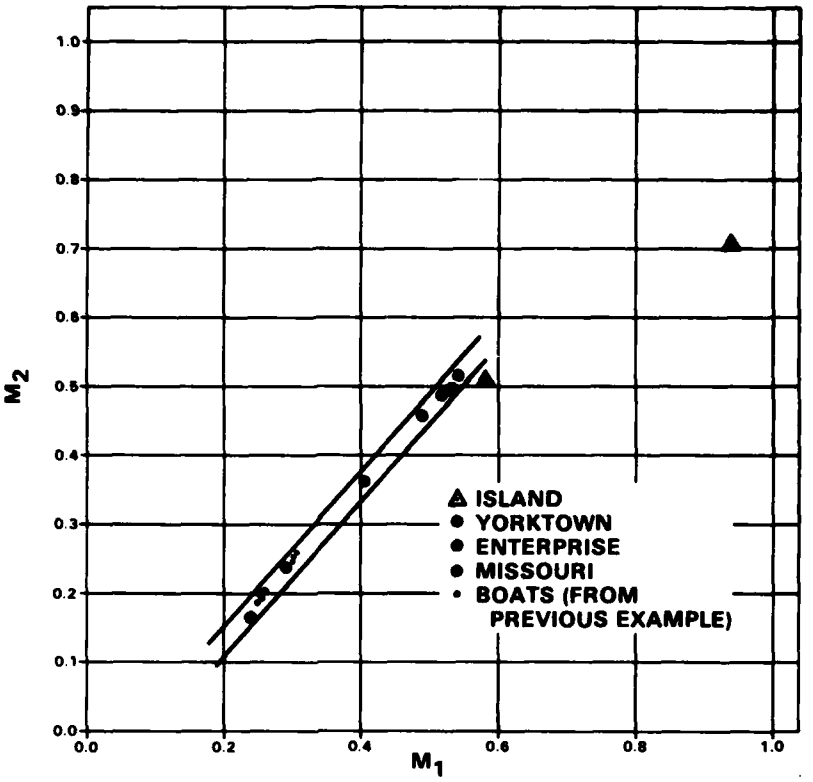


Figure 22. Sea Scene as Points in Feature Space

The short-list algorithm used here is not intended to be the last word in picture parsing, but rather a step along the way to a really efficient algorithm. This aspect of target recognition should be further investigated as well.

REFERENCES

1. Ming-Kuei Hu, "Visual Pattern Recognition by Moment Invariants," *IRE Trans. Inform. Theory*, Vol IT-8 (February 1962), pp 179–187.

2. Ming-Kuei Hu, *et al.*, *Identification of Three-Dimensional Aircraft by Their Two-Dimensional Optical Images*, Rome Air Development Center RADC-TR-75-201, Griffis Air Force Base, N.Y.

3. L. B. Scott and K. Preston, "Apparatus for Counting Irregularly Shaped Objects," U.S. Patent 3,408,485, 1968.

4. A. K. Agrawala and A. V. Kulkarni, "A Sequential Approach to the Extraction of Shape Features," *Computer Graphics and Image Processing*, Vol 6, No. 6 (December 1977), pp 538–557.

5. C. Northcote Parkenson, *Parkenson's Law.* Boston: Houghton Mifflin Co., 1957.

6. S. A. Dudani, K. J. Breeding, R. B. McGhee, "Aircraft Identification by Moment Invariants," *IEEE Transactions on Computers*, Vol C-26 (January 1977), pp 39–45.

APPENDIX A. PROOF OF MOMENT INVARIANTS

Show that the functions

$$M_1 = (\mu_{20} + \mu_{02})^2$$

$$M_2 = (\mu_{20} - \mu_{02})^2 + 4\mu_{11}^2$$

are invariant with respect to notation.

Let the primed variables M_1', M_2', x_i', y_i' represent a pattern made up of N pixels (x_i, y_i) transformed by

$$\begin{bmatrix} x_i' \\ y_i' \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix}.$$

Then

$$M_1' = \sum_{i=1}^{N} \left[x_i'^2 + y_i'^2\right]^2$$

$$= \sum_{i=1}^{N} \left[(x_i \cos\theta + y_i \sin\theta)^2 + (-x \sin\theta + y_i \cos\theta)^2\right]^2$$

$$= \sum_{i=1}^{N} \left[x_i^2 \cos^2\theta + 2x_i y_i \sin\theta \cos\theta + y_i^2 \sin^2\theta \right.$$

$$\left. + x_i^2 \sin^2\theta - 2x_i y_i \sin\theta \cos\theta + y_i^2 \cos^2\theta\right]^2$$

$$= \sum_{i=1}^{N} \left[x_i^2 (\sin^2\theta + \cos^2\theta) + y_i^2 (\sin^2\theta + \cos^2\theta)\right]^2$$

$$= \sum_{i=1}^{N} \left[x_i^2 + y_i^2\right] = M_1 .$$

Likewise

$$M_2' = \sum_{i=1}^{N} \left[(x_i \cos\theta + y_i \sin\theta)^2 - (-x_i \sin\theta + y_i \cos\theta)^2\right]^2$$

$$+4[(x_i \cos\theta + y_i \sin\theta)(-x_i \sin\theta + y_i \cos\theta)]^2$$

$$= \sum_{i=1}^{N} [(x_i^2 - y_i^2)(\cos^2\theta - \sin^2\theta) + 4x_i y_i \sin\theta \cos\theta]^2$$

$$+ 4\left[(y_i^2 - x_i^2 \sin\theta \cos\theta + x_i y_i (\cos^2\theta - \sin^2\theta)\right]^2$$

$$= \sum_{i=1}^{N} [(x_i^2 - y_i^2)\cos 2\theta + 2x_i y_i \sin 2\theta]^2$$

$$+ 4\left[-\frac{x_i^2 - y_i^2}{2} \sin 2\theta + x_i y_i \cos 2\theta\right]^2$$

$$= \sum_{i=1}^{N} x_i^2 - y_i^2 + 4x_i y_i$$

$$= M_2 .$$

APPENDIX B. FORTRAN PROGRAM

The following Fortran program simulates a video signal processor incorporating the short list algorithm and moment invariant computation. It accepts pixel data on cards. Each input image consists of a background of blanks (or zeros) with patterns of ones to be transformed and recognized. The images are 64 pixels in width and may have any length. Pixels are punched in columns 1 through 64 as blanks or (1)s. Column 65 is punched (1) if the card does not represent the bottom of the image. A card representing the bottom of the image has column 65 blank or zero and causes the program to process the image and print out (1) the picture in blank or (*) format, (2) the raw moments for each object in the field of view, and (3) the moment invariants. If a card representing the bottom of an image has a 1 punched in column 66, the program expects another image to follow in the deck. Otherwise it stops.

```
      PROGRAM PARSE (INPUT,OUTPUT)
C-----------------------------------------------------------------C
C                                                                 C
C          PROGRAM TO PARSE BINARY IMAGES INTO ISOLATED           C
C                                                                 C
C              SILHOUETTS AND COMPUTE THEIR MOMENTS               C
C                                                                 C
C-----------------------------------------------------------------C
C
      DIMENSION KARD(64), IPL(64), IPR(64), MEM(45,13), FONT(45)
      DIMENSION  ISHOT(45)
      DIMENSION FEAT(10)
      EQUIVALENCE (FEAT(1),F00),(FEAT(2),F01,YBAR),(FEAT(3),F02),
     1(FEAT(4),F03),(FEAT(5),F10,XBAR),(FEAT(6),F11),(FEAT(7),F12)
     2(FEAT(8),F20),(FEAT(9),F21),(FEAT(10),F30)
      INTEGER FONT
C
C     ...
C     READ FONT
C     ...
      READ 700, (FONT(I),I=1,45)
C     ...
C     PRINT HEADERS
C     ...
    3 PRINT 701
      PRINT 702
      PRINT 703
      PRINT 704
      PRINT 705
C     ...
C     INITIALIZE PICTURE PROCESSING
C     ...
      LINE=0
      DO 1 I=1,45
      DO 1 J=1,13
    1 MEM(I,J)=0
      DO 2 K=1,64
    2 IPL(K)=0
C     ...
C     READ AND PROCESS ONE LINE OF VIDEO
C     ...
   10 IP=0
      LINE=LINE+1
      READ 706,(KARD(K),K=1,64),NEXT,MORE
C
C
      DO 11 LR=1,64
      K=LR
      LODD=LINE.AND.1
      IF (LODD.EQ.0) K=65-K
      IA=IPL(K)
      IPL(K)=0
      IPR(K)=1H
      MAR=0
      IF(KARD(K).EQ.0) GO TO 11
C     ...
C     GIVEN *
```

```
      C     ...
            IF(IA.EQ.0.AND.IP.EQ.0) GO TO 12
   60       IF(IA.EQ.0.AND.IP.NE.0) GO TO 13
            IF(IA.NE.0.AND.IP.EQ.0) GO TO 14
            IF(IA.EQ.IP)            GO TO 13
            IF(IA.LT.IP)            GO TO 15
      C     ...
   65    C  SET  P INTO A(REF)
      C     ...
            MEM(IA,2)=IP
            MEM(IP,1)=MEM(IP,1)+1
      C     ...
   70    C  ASSIGN P , SET LINE FLAG
      C     ...
      13    MAR=IP
            MEM(MAR,3)=1
            GO TO 16
   75    C  ...
      C     SET  A INTO P(REF)
      C     ...
      15    MEM(IP,2)=IA
            MEM(IA,1)=MEM(IA,1)+1
   80       ...
      C     ASSIGN A , SET LINE FLAG
      C     ...
      14    MAR=IA
            MEM(MAR,3)=1
   85       GO TO 16
      C     ...
      C     SELECT NEW CHARACTER
      C     ...
      12    IFIND=1
   90       DO 120 ICK=1,45
            IF(MEM(IFIND,1).EQ.0)GO TO 121
            IFIND=IFIND+1
      120   CONTINUE
            PRINT 707
   95       GO TO 60
      121   MAR=IFIND
            MEM(MAR,1)=1
      C     ...
      C     INTEGRATE
  100       ...
      16    IPL(K)=MAR
            IPR(K)=FONT(MAR)
            MEM(MAR,3)=1
            MEM(MAR,4)=MEM(MAR,4)+1
  105       MEM(MAR,5)=MEM(MAR,5)+LINE
            MEM(MAR,6)=MEM(MAR,6)+LINE**2
            MEM(MAR,7)=MEM(MAR,7)+LINE**3
            MEM(MAR,8)=MEM(MAR,8)+K
            MEM(MAR,9)=MEM(MAR,9)+K*LINE
  110       MEM(MAR,10)=MEM(MAR,10)+K*LINE**2
            MEM(MAR,11)=MEM(MAR,11)+K**2
            MEM(MAR,12)=MEM(MAR,12)+K**2*LINE
            MEM(MAR,13)=MEM(MAR,13)+K**3
      11    IP=MAR
```

```
115          C     ...
             C     ...
             C     PURGE EXTRANEOUS ENTRIES
             C     ...
                   DO 1100 IS=1,45
120          1100  ISHOT(IS)=1H
                   IS=1
                   DO 100 IFWD=1,45
                   IREV=46-IFWD
                   IF(MEM(IREV,3).EQ.1) GO TO 102
125                IF(NEXT.NE.0.AND.MEM(IREV,1).NE.1) GO TO 100
                   IF(NEXT.EQ.0.AND.MEM(IREV,1).EQ.0) GO TO 100
                   IF(MEM(IREV,2).NE.0) GO TO 103
                   IF(MEM(IREV,4).GE.16) GO TO 100
             104   IRIP=MEM(IREV,2)
130                ISHOT(IS)=FONT(IREV)
                   IS=IS+1
                   DO 105 IW=1,13
             105   MEM(IREV,IW)=0
                   GO TO 100
135          102   MEM(IREV,3)=0
                   GO TO 100
             103   IREF=MEM(IREV,2)
                   DO 107 IW=4,13
             107   MEM(IREF,IW)=MEM(IREF,IW)+MEM(IREV,IW)
140                IF(MEM(IREF,1).GT.1) MEM(IREF,1)=MEM(IREF,1)-1
                   IF(MEM(IREF,1).LE.1) ISHOT(45)=1H+
                   GO TO 104
             100   CONTINUE
             C     ...
145          C     FINISH LINE , PRINT TRACE
             C     ...
                   PRINT 700,LINE,(IPR(II),II=1,64),(ISHOT(III),III=1,45)
             C     ...
             C     TEST FOR END OF DECK
150          C     ...
                   IF(NEXT.NE.0) GO TO 10
              60   PRINT 705
             C     ...
             C     PRINT MEMORY , CLEAR DENSE PRINT MODE
155          C     ...
                   PRINT 710
                   PRINT 711
                   DO 70 LOC=1,45
                   IF(MEM(LOC,1).NE.0)PRINT 712,LOC,FONT(LOC),(MEM(LOC,IW),IW=1,13)
160           70   CONTINUE
             C     ...
             C     PRINT SECOND PHASE HEADERS
             C     ...
                   PRINT 720
165                PRINT 721
             C     ...
             C     LOOP THRU MEMORY, COMPUTE FEATURE VECTOR
             C     ...
                   DO  56  L=1,45
170                IF (MEM(L,1).EQ.0) GO TO 56
                   IF (MEM(L,4).LT.16) GO TO 56
```

```
C     ...
C     CONVERT SUMS TO ORDINARY MOMENTS
C     ...
      DO 55 IW=1,10
      IWW=IW+3
   55 FEAT(IW)=MEM(L,IWW)
C     ...
C     COMPUTE 3-RD ORDER CENTRAL MOMENTS
C     ...
      F03= F03-3.0*F02*(F01/F00)+2.0*F00*(F01/F00)**3
      F30= F30-3.0*F20*(F10/F00)+2.0*F00*(F10/F00)**3
      F12= F12-F02*(F10/F00)-2.0*F11*(F01/F00)+2.0*F10*(F01/F00)**2
      F21= F21-F20*(F01/F00)-2.0*F11*(F10/F00)+2.0*F01*(F10/F00)**2
C     ...
C     COMPUTE 2-ND ORDER CENTRAL MOMENTS
C     ...
      F11=F11-F00*(F10/F00)*(F01/F00)
      F02=F02-F00*(F01/F00)**2
      F20=F20-F00*(F10/F00)**2
C     ...
C     PRINT
C     ...
      PRINT 723
      PRINT 722,L,FONT(L),(FEAT(IW),IW=1,10)
C     ...
C     NORMALIZE W.R.T. SIZE
C     ...
      XBAR=F10/F00
      YBAR=F01/F00
      F002=F00**2
      F052=F00**2.5
      F02=F02/F002
      F03=F03/F052
      F11=F11/F002
      F12=F12/F052
      F20=F20/F002
      F21=F21/F052
      F30=F30/F052
      PRINT 725,          (FEAT(IW),IW=1,10)
C     ...
C     COMPUTE THREE MOMENT INVARIENTS
C     ...
      VAR1=F20+F02
      VAR2= (F20-F02)**2+4.0*F11**2
      VAR3= (F30-3.0*F12)**2+(3.0*F21-F03)**2
      VAR4= (F30+F12)**2+(F21+F03)**2
C     ...
      PRINT 724,VAR1,VAR2,VAR3,VAR4
   56 CONTINUE
      IF(MORE.NE.0) GO TO 3
      STOP
  700 FORMAT (45A1)
  701 FORMAT (1H1/1HT)
  702 FORMAT (23H          PROGRAM PARSE)
  703 FORMAT (1H0,18X,10(1H1),10(1H2),10(1H3),10(1H4),10(1H5),5(1H6))
  704 FORMAT (10X,6(10H1234567890),4H1234)
  705 FORMAT (9X,66(1H*))
```

```
                706   FORMAT (66I1)
    230         707   FORMAT (23H0  NO MORE MEMORY SPACE)
                708   FORMAT(I9,1H*,64A1,1H*,2X,45A1)
                C
                C
                710   FORMAT (1H1/1HS)
    235         711   FORMAT (16H           MEMORY/                                  60H
                     1 LOC CH  U REF L          M00       M01       M02       M03,  60H
                     2       M10       M11       M12       M20       M21       M30)
                712   FORMAT (I4,2X,A1,3I3,4X,10I10)
                720   FORMAT(40H1  MOMENT INVARIANT  COMPUTATION  PHASE )
    240         721   FORMAT(50H0 LOC      CHARACTER       F00       F01       F02 , 60H
                     1       F03       F10       F11       F12       F20       F21 , 10H
                     2       F30)
                722   FORMAT(I4,11X,A1,4X,10F10.3)
                723   FORMAT(1H )
    245         724   FORMAT (3X, 7HVAR1-4=,4E14.5)
                725   FORMAT(20H    SIZE NORNALIZE  ,10F10.3)
                      END
```

SYMBOLIC REFERENCE MAP (R=1)

ENTRY POINTS
4112 PARSE

VARIABLES	SN	TYPE	RELOCATION				
5057	FEAT	REAL	ARRAY	6582	FONT	INTEGER	ARRAY
5057	F00	REAL		5051	F002	REAL	
5060	F01	REAL		5061	F02	REAL	
5062	F03	REAL		5052	F052	REAL	
5063	F10	REAL		5064	F11	REAL	
5065	F12	REAL		5066	F20	REAL	
5067	F21	REAL		5070	F30	REAL	
5021	I	INTEGER		5032	IA	INTEGER	
5035	ICK	INTEGER		5034	IFIND	INTEGER	
5037	IFWD	INTEGER		5044	II	INTEGER	
5045	III	INTEGER		5025	IP	INTEGER	
5171	IPL	INTEGER	ARRAY	5271	IPR	INTEGER	ARRAY
5043	IREF	INTEGER		5040	IREV	INTEGER	
5041	IRIP	INTEGER		5036	IS	INTEGER	
6557	ISHOT	INTEGER	ARRAY	5042	IW	INTEGER	
5050	IWW	INTEGER		5023	J	INTEGER	
5024	K	INTEGER		5071	KARD	INTEGER	ARRAY
5047	L	INTEGER		5022	LINE	INTEGER	
5046	LOC	INTEGER		5031	LOOD	INTEGER	
5030	LR	INTEGER		5033	MAR	INTEGER	
5371	MEM	INTEGER	ARRAY	5027	MORE	INTEGER	
5026	NEXT	INTEGER		5053	VAR1	REAL	
5054	VAR2	REAL		5055	VAR3	REAL	
5056	VAR4	REAL		5063	XBAR	REAL	
5060	YBAR	REAL					

APPENDIX C. BASIC LANGUAGE PROGRAMS

The following two BASIC language programs for the Tektronix 4051 were written by Richard Ely. The first determines (interactively with the user) which portion of the image is to be stored. It then places the pixel array in a cartridge file. The second program prints out the contents of the file as a reconstructed printer image from which the numerical values assigned the pixels are evident.

```
2000 INIT
2005 PAGE
2010 DIM A0(129),A1(129),B(1024),A2(22),H(64)
2020 A0=0
2030 B=0
2040 A1=252
2050 REM 'THIS SECTION INPUTS THE COORDINATES OF THE DESIRED IMAGE ON
2060 REM  THE TV SCREEN.  IT PUTS A CROSSHAIR ON THIS POSITION SO YOU
2070 REM  CAN SEE IF IT IS THE RIGHT LOCATION.'
2080 PRINT 'ENTER THE COORDINATES OF THE CENTER POINT.'
2090 PRINT 'Y=?, (16 TO 496)'
2100 INPUT Y
2110 PRINT 'X=?, (16 TO 496)'
2120 INPUT X
2130 REM
2140 REM 'THE CENTER POINT OF THE CROSSHAIRS IS NOT ALLOWED TO GET
2150 REM  CLOSER THAN 16 TO THE EDGES OF THE PICTURE.  THE VALUES OF
2160 REM  X AND Y ARE CHANGED IF NECESSARY TO SATISFY THIS CONDITION.'
2170 REM
2180 IF Y=>16 THEN 2200
2190 Y=16
2200 IF Y<=496 THEN 2220
2210 Y=496
2220 IF X=>16 THEN 2240
2230 X=16
2240 IF X<=496 THEN 2300
2250 X=496
2260 REM
2270 REM 'THE STARTING POINT FOR THE CROSSHAIRS IS 10 LINES ABOVE THE
2280 REM  CENTER POINT.'
2290 REM
2300 N1=(Y-10)*512+(X-1)
2310 REM
2320 REM '21 VERTICAL LINES ARE READ OUT AND WRITTEN INTO BY THE FOL-
2330 REM  LOWING INSTRUCTIONS TO FORM THE VERTICAL CROSSHAIR.'
2340 REM
2350 FOR I=1 TO 21
2360 N=N1
2370 GOSUB 5020
2380 K2=3*(I-1)
2390 REM
2400 REM 'THE FOLLOWING TELLS THE QUANTEX IT IS GOING TO OUTPUT DATA
2410 REM  STARTING AT LOCATION N(G,F,E,D,C) AND THAT THERE WILL BE 5
2420 REM  WORDS SENT OUT(37,48,64,80,96).'
2430 REM
2440 WBYTE @36:-1,0,1,61,34,2,G,F,E,D,C,3,37,48,64,80,96,15
2450 REM
2460 REM 'THE QUANTEX IS FORMALLY ADDRESSED AS A TALKER.'
2470 REM
2480 WBYTE @68:
2490 REM
2500 REM 'THE 4051 RECEIVES THE DATA IN THE 'B' ARRAY.'
2510 REM
2520 RBYTE B(K2+1),B(K2+2),B(K2+3),B(K2+4)
2530 REM
2540 REM 'THE QUANTEX IS UNTALKED AND THEN ADDRESSED AS A LISTENER.  IT
2550 REM  RECEIVES THE DATA 0,252, AND 0.'
2560 REM
2570 WBYTE @95,36:-1,0,1,57,34,2,G,F,E,D,C,15,0,252,0,-1
2580 N1=N1+512
2590 NEXT I
2600 REM
2610 REM 'THE HORIZONTAL BAR IS ADDED TO THE CROSSHAIR AFTER READING
2620 REM  OUT THE PICTURE INFORMATION.'
2630 REM
2640 N1=(Y-1)*512+(X-10)
```

```
2650 FOR I=1 TO 3
2660 N=N1
2670 GOSUB 5020
2680 WBYTE @36:-1,0,1,61,34,2,G,F,E,D,C,3,39,49,64,80,96,15
2690 WBYTE @68:
2700 RBYTE A2
2710 K2=63+21*(I-1)
2720 FOR J=1 TO 21
2730 B(K2+J)=A2(J)
2740 NEXT J
2750 DIM A2(21)
2760 A2=0
2770 WBYTE @95,36:-1,0,1,57,34,2,G,F,E,D,C,15,A2,-1
2780 N1=N1+512
2790 NEXT I
2800 N=N1-1024
2810 GOSUB 5020
2820 A2=252
2830 WBYTE @95,36:-1,0,1,57,34,2,G,F,E,D,C,15,A2,-1
2840 REM
2850 REM 'IF THE CROSSHAIR IS NOT IN THE RIGHT LOCATION, YOU CAN MOVE
2860 REM  IT.  IN EITHER CASE IT WILL WRITE THE ORIGINAL PICTURE BACK IN
2870 REM  TO THE QUANTEX BEFORE EITHER GOING BACK TO THE BEGINNING OR
2880 REM  LOADING THE PICTURE ONTO A CASSETTE TAPE FILE.'
2890 REM
2900 PRINT 'IS THIS THE POINT YOU WANT?'
2910 INPUT A$
2970 N1=(Y-1)*512+(X-10)
2980 FOR I=1 TO 3
2990 N=N1
3000 GOSUB 5020
3010 K2=63+21*(I-1)
3020 FOR J=1 TO 21
3030 A2(J)=B(K2+J)
3040 NEXT J
3050 WBYTE @36:-1,0,1,57,34,2,G,F,E,D,C,15,A2,-1
3060 N1=N1+512
3070 NEXT I
3080 N1=(Y-10)*512+(X-1)
3090 FOR I=1 TO 21
3100 N=N1
3110 GOSUB 5020
3120 K2=3*(I-1)
3130 WBYTE @36:-1,0,1,57,34,2,G,F,E,D,C,15,B(K2+1),B(K2+2),B(K2+3),-1
3140 N1=N1+512
3150 NEXT I
3160 IF A$='Y' THEN 3180
3170 GO TO 2010
3180 REM
3190 REM '128 LINES OF 128 PIXELS EACH ARE READ OUT FROM THE QUANTEX.
3200 REM  A SQUARE OF FOUR PIXELS AREADDED TOGETHER AND DIVIDED BY FOUR
3210 REM  TO FORM A NEW DATA POINT.  THIS REDUCES THE ARRAY TO 64X64
3220 REM  PIXELS.  THESE PIXELS ARE PACKED FOUR TO ONE EIGHT DIGIT
3230 REM  INTEGER.  THUS THE ARRAY IS PACKED INTO 1024 NUMBERS.
3240 REM
3241 PRINT 'A 128X128 PIXEL IMAGE WILL NOW BE READ OUT.  YOU CAN'
3242 PRINT 'HAVE THIS IMAGE REDUCED TO 64X64 PIXELS BY TYPING R. '
3243 PRINT 'IF YOU WANT THE FULL IMAGE TYPE F.'
3244 INPUT F$
3245 IF F$='F' THEN 4000
3250 H=0
3260 N1=(Y-64)*512+(X-64)
3270 FOR I=1 TO 64
3280 N=N1
3290 GOSUB 5020
3300 WBYTE @36:-1,0,1,61,34,2,G,F,E,D,C,3,36,56,64,80,96,15
```

```
3310 WBYTE @68:
3320 RBYTE A0
3330 N=N1+512
3340 GOSUB 5020
3350 WBYTE @36:-1,0,1,61,34,2,G,F,E,D,C,3,36,56,64,80,96,15
3360 WBYTE @68:
3370 RBYTE A1
3380 K2=(I-1)*16
3390 FOR J=1 TO 16
3400 K3=(J-1)*8
3410 X1=INT((A0(K3+1)+A0(K3+2)+A1(K3+1)+A1(K3+2))/16)
3420 X2=INT((A0(K3+3)+A0(K3+4)+A1(K3+3)+A1(K3+4))/16)
3430 X3=INT((A0(K3+5)+A0(K3+6)+A1(K3+5)+A1(K3+6))/16)
3440 X4=INT((A0(K3+7)+A0(K3+8)+A1(K3+7)+A1(K3+8))/16)
3450 B(K2+J)=X1*1000000+X2*10000+X3*100+X4
3460 H(X1+1)=H(X1+1)+1
3470 H(X2+1)=H(X2+1)+1
3480 H(X3+1)=H(X3+1)+1
3490 H(X4+1)=H(X4+1)+1
3500 NEXT J
3510 IF I<3 OR I>62 THEN 3580
3520 A0(1)=252
3530 A0(2)=252
3540 A0(127)=252
3550 A0(128)=252
3560 A0(129)=-1
3570 GO TO 3600
3580 A0=252
3590 A0(129)=-1
3600 WBYTE @36:-1,0,1,57,34,2,G,F,E,D,C,15,A0
3610 N1=N1+1024
3620 NEXT I
3621 PRINT 'WHAT TAPE FILE DO YOU WANT TO STORE THE IMAGE IN?  IT'
3622 PRINT 'MUST BE AT LEAST 44000 BYTES LONG.'
3623 INPUT A5
3624 PRINT 'INSERT THE TAPE AND HIT RETURN.'
3625 INPUT F$
3630 FIND A5
3640 WRITE 1024,16,64,H,B
3650 END
4000 REM '128 BY 128 PIXELS ARE READ OUT AND STORED IN A FILE ON
4010 REM  TAPE.'
4020 PRINT 'WHAT TAPE FILE DO YOU WANT TO STORE THE IMAGE IN?  IT'
4021 PRINT 'MUST BE AT LEAST 88000 BYTES LONG.'
4022 INPUT A5
4023 PRINT 'INSERT THE TAPE AND HIT RETURN.
4024 INPUT F$
4025 DELETE A$,A0,A1,A2,D
4030 DIM A0(129),A1(129),A2(129),M(32),A$(365),B$(365)
4031 A=1
4032 CALL 'UNIT',A
4033 CALL 'MOUNT',A,A$
4035 OPEN '@IMAGE/ONE','G',1,'F',B$
4036 PRINT B$
4037 CALL 'REWIND',1
4040 A0=0
4050 A1=0
4060 A2=0
4260 N1=(Y-64)*512+(X-64)
4270 FOR I=1 TO 128
4285 N=N1
4290 GOSUB 5020
4300 WBYTE @36:-1,0,1,61,34,2,G,F,E,D,C,3,36,56,64,80,96,15
4310 WBYTE @68:
4320 RBYTE A2
4330 A0=A1
```

```
4340 A1=A2
4350 L=1
4390 FOR J=1 TO 127 STEP 4
4400 X1=INT(A2(J)*1^9)
4410 X2=INT(A2(J+1)*1^6)
4420 X3=INT(A2(J+2)*1^3)
4430 X4=INT(A2(J+3))
4440 M(L)=X1+X2+X3+X4
4450 L=L+1
4590 NEXT J
4610 IF I<3 OR I>126 THEN 4680
4620 A2(1)=252
4630 A2(2)=252
4640 A2(127)=252
4650 A2(128)=252
4660 A2(129)=-1
4670 GO TO 4700
4680 A2=252
4690 A2(129)=-1
4700 WBYTE @36:-1,0,1,57,34,2,G,F,E,D,C,15,A2
4701 WBYTE @63,95:
4705 WRITE #1:M(1),M(2),M(3),M(4),M(5),M(6),M(7),M(8),M(9),M(10)
4706 WRITE #1:M(11),M(12),M(13),M(14),M(15),M(16),M(17),M(18),M(19)
4707 WRITE #1:M(20),M(21),M(22),M(23),M(24),M(25),M(26),M(27),M(28)
4708 WRITE #1:M(29),M(30),M(31),M(32)
4709 CLOSE 1
4710 OPEN '@IMAGE/ONE':1,'U',B%
4715 N1=N1+512
4720 NEXT I
4730 CLOSE 1
4740 CALL 'DISMOUNT',1
4750 END
5000 REM 'THIS SUBROUTINE CALCULATES THE STARTING POINT IN THE QUANTEX
5010 REM  AND PUTS IN THE FORMAT IT UNDERSTANDS.'
5020 C=INT(N/65536)
5030 N=N-C*65536
5040 D=INT(N/4096)
5050 N=N-D*4096
5060 E=INT(N/256)
5070 N=N-E*256
5080 F=INT(N/16)
5090 G=N-F*16
5100 C=C+96
5110 D=D+80
5120 E=E+64
5130 F=F+48
5140 G=G+32
5150 RETURN
5160 END
```

```
100 INIT
110 PAGE
120 DIM H(64),B$(64),C$(64)
130 PRINT @51:'L'
140 PRINT 'WHAT IMAGE FILE DO YOU WANT TO LOOK AT?'
150 INPUT A5
160 PRINT 'INSERT THE TAPE THAT CONTAINES THE IMAGE AND PRESS'
170 PRINT 'RETURN'
180 INPUT A6
181 PRINT 'WHAT DO YOU WANT TO TITLE THE PICTURE?'
182 INPUT D$
190 FIND A5
200 READ @33:X,X8,Y8,H
210 T3=X8*Y8
220 DIM B(T3)
230 READ @33:B
240 REM 'FINDS THE MIN AND MAX PIXEL VALUES.'
250 I=1
260 IF H(I)>0 THEN 300
270 I=I+1
280 IF I>64 THEN 370
290 GO TO 260
300 M1=I-1
310 I=64
320 IF H(I)>0 THEN 350
330 I=I-1
340 GO TO 320
350 M2=I-1
360 PRINT @51:M1,M2,D$,'J'
370 Y=9.99/(M2-M1)
380 FOR J=1 TO 16
390 FOR L=1 TO 4
400 B$=''
410 C$=''
420 K2=100^(4-L)
430 I=64
440 K=J+(I-1)*16
450 X=INT(B(K)/K2)
460 B(K)=B(K)-X*K2
470 N=INT((X-M1)*Y)
480 GOSUB 570
490 I=I-1
500 IF I=0 THEN 520
510 GO TO 440
520 PRINT @51:B$
530 PRINT @51:C$,'J'
540 NEXT L
550 NEXT J
560 END
570 GOSUB N+1 OF 860,830,800,770,740,710,680,650,620,590
580 RETURN
590 B$=B$$' '
600 C$=C$$' '
610 RETURN
620 B$=B$$'.'
630 C$=C$$' '
640 RETURN
650 B$=B$$':'
660 C$=C$$' '
670 RETURN
680 B$=B$$'I'
690 C$=C$$' '
700 RETURN
710 B$=B$$'U'
720 C$=C$$' '
730 RETURN
```

```
740 B$=B$$"0"
750 C$=C$$" "
760 RETURN
770 B$=B$$"0"
780 C$=C$$"-"
790 RETURN
800 B$=B$$"0"
810 C$=C$$"+"
820 RETURN
830 B$=B$$"0"
840 C$=C$$"*"
850 RETURN
860 B$=B$$"0"
870 C$=C$$"#"
880 RETURN
```

DISTRIBUTION

Defense Technical Information Center
Cameron Station
Alexandria, VA 22314 (12)

Library of Congress
Washington, DC 20540
ATTN: Gift and Exchange Division (4)

Local:

E31 (GIDEP)
E41
F14 (Miller)
N11 (Rowan, 2-117)
N14 (Gray, 30-105)
N14 (Moscar, 30-325)
K60 (Shelton)
K61 (Olsen)
K61 (Hilton) (20)
X210 (2)
X211 (2)

IED Panel:

C2 (Veazey)
E1A (McKnight, 4-176)
E21 (Mebane, 20-231)
F56 (Puglielli)
G41 (Walchak, 20-224)
G501 (Culbertson)
K04 (Schindel)
K74 (Thombs)
N43 (Pater)
R01 (Nicholson, 1-258)
R45 (Savage, 1-017)
U02 (Fanjoy, 1-232)