AUTOMATED READING OF VIDEO TAPE (U)

NOV 80 R G MACHUCA

UNCLASSIFIED

STEWS-10-80-2

END

DATE
AD-981

DTIC
Destroy this report when no longer needed. Do not return it to the originator.

DISCLAIMER

The findings of this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.
This report describes the concepts and software which are used in a prototype videotape reaper being developed at WSMR. The goal is to read from one frame of video the position of the target and the three angular rotations of the target. This is done by first making a contour of the target from a frame of video. The Fourier descriptors of the contour are then computed and normalized.
ACKNOWLEDGEMENTS

The author would like to acknowledge those who helped with this report. Ms. P. Smith assisted with the programming. Mr. J. Wise allowed us the use of the laboratory, and Mr. C. Klaassen, as Systems Manager, was very helpful in making the equipment available to us. Ms. H. Essary prepared the final report, with help from Mr. M. Ramos in preparing the figures.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>SYSTEM DESCRIPTION</td>
<td>1</td>
</tr>
<tr>
<td>STRUCTURE OF THE PREPROCESSING SOFTWARE</td>
<td>9</td>
</tr>
<tr>
<td>CONSTRUCTION OF A CONTOUR FROM THE MOMENT FILE</td>
<td>16</td>
</tr>
<tr>
<td>FOURIER DESCRIPTOR METHODS</td>
<td>42</td>
</tr>
<tr>
<td>Appendix A. FINDING EDGES IN NOISY SCENES</td>
<td>A-1</td>
</tr>
<tr>
<td>Section 1. Edges from Moments</td>
<td>A-1</td>
</tr>
<tr>
<td>Section 2. Second Order Edges</td>
<td>A-5</td>
</tr>
<tr>
<td>Section 3. Algorithms for Implementation</td>
<td>A-6</td>
</tr>
<tr>
<td>a. Calculation of Moment</td>
<td>A-6</td>
</tr>
<tr>
<td>b. Calculation of the Rotation</td>
<td>A-8</td>
</tr>
<tr>
<td>Section 4. Evaluation</td>
<td>A-9</td>
</tr>
</tbody>
</table>
# LIST OF ILLUSTRATIONS

<table>
<thead>
<tr>
<th>Figures</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1a.</td>
<td>Digitized video images of F102</td>
<td>3</td>
</tr>
<tr>
<td>1b.</td>
<td>Result of processing original with an edge detector</td>
<td>3</td>
</tr>
<tr>
<td>1c.</td>
<td>Contour of F102</td>
<td>3</td>
</tr>
<tr>
<td>2.</td>
<td>Line segments and associated segment numbers</td>
<td>4</td>
</tr>
<tr>
<td>3.</td>
<td>Library of contours of F102. This library was generated from a computer made three-dimensional model</td>
<td>5</td>
</tr>
<tr>
<td>4.</td>
<td>Some contours found by the computer before being corrected by the operator</td>
<td>6</td>
</tr>
<tr>
<td>5.</td>
<td>Contours of Figure 4 after being processed by the operator</td>
<td>7</td>
</tr>
<tr>
<td>6.</td>
<td>The best match found by the computer</td>
<td>8</td>
</tr>
<tr>
<td>7.</td>
<td>Structure of the preprocessing software</td>
<td>10</td>
</tr>
<tr>
<td>8.</td>
<td>Generic hardware model for preprocessing hardware</td>
<td>15</td>
</tr>
<tr>
<td>9a.</td>
<td>F102 flying by a mountain</td>
<td>17</td>
</tr>
<tr>
<td>9b.</td>
<td>F102 flying in front of a mountain</td>
<td>17</td>
</tr>
<tr>
<td>9c.</td>
<td>Hawk missile</td>
<td>17</td>
</tr>
<tr>
<td>9d.</td>
<td>F102, sunspots on wings and nose</td>
<td>17</td>
</tr>
<tr>
<td>10a.</td>
<td>Moment file of F102 flying by a mountain</td>
<td>18</td>
</tr>
<tr>
<td>10b.</td>
<td>Moment file of F102 flying in front of a mountain</td>
<td>18</td>
</tr>
<tr>
<td>10c.</td>
<td>Moment file of Hawk missile</td>
<td>18</td>
</tr>
<tr>
<td>10d.</td>
<td>Moment file of F102, sunspots on wings and nose</td>
<td>18</td>
</tr>
<tr>
<td>11.</td>
<td>Raw histogram Data (I) from moment file. Integrated Data (II) from raw histograms</td>
<td>19</td>
</tr>
<tr>
<td>12a.</td>
<td>Contour found from Moment file (Fig. 10a)</td>
<td>20</td>
</tr>
<tr>
<td>12b.</td>
<td>Contour found from Moment file (Fig. 10b)</td>
<td>20</td>
</tr>
<tr>
<td>12c.</td>
<td>Contour found from Moment file (Fig. 10c)</td>
<td>20</td>
</tr>
<tr>
<td>12d.</td>
<td>Contour found from Moment file (Fig. 10d)</td>
<td>20</td>
</tr>
<tr>
<td>13.</td>
<td>Two-dimensional number patterns and their assigned geometrical directions</td>
<td>24</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>---------</td>
<td>-----------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>14.</td>
<td>Examples of geometric relations for various predicates, the geometric situation when (a) $P = \text{true}$, and (b) $Q = \text{true}$</td>
<td>27</td>
</tr>
<tr>
<td>15.</td>
<td>Data structures (a), (b) and operations with their corresponding (c), (d) geometric structures and geometric operations</td>
<td>34</td>
</tr>
<tr>
<td>16.</td>
<td>Geometric criteria for the continuation of a polygonal segment</td>
<td>36</td>
</tr>
<tr>
<td>17.</td>
<td>Polygon before closing by distance measure</td>
<td>37</td>
</tr>
<tr>
<td>18.</td>
<td>Polygon after closing by distance measure</td>
<td>38</td>
</tr>
<tr>
<td>19.</td>
<td>Polygons and corrections entered by the operator</td>
<td>43</td>
</tr>
<tr>
<td>20.</td>
<td>New polygons obtained from corrected polygons of Figure 19</td>
<td>44</td>
</tr>
<tr>
<td>21.</td>
<td>Three different contours of an F102 from video taken at WSMR</td>
<td>45</td>
</tr>
<tr>
<td>22.</td>
<td>Three basic geometrical shapes and their Fourier coefficients</td>
<td>46</td>
</tr>
<tr>
<td>23.</td>
<td>Contours generated by functions of the type $Z(t) = \exp(it) + \frac{i}{L} \exp(itL)$</td>
<td>49</td>
</tr>
<tr>
<td>A-1.</td>
<td>Example center of mass vectors</td>
<td>A-2</td>
</tr>
<tr>
<td>A-2.</td>
<td>Rocket and results of processing</td>
<td>A-3</td>
</tr>
<tr>
<td>A-3.</td>
<td>A curve $\Gamma$ and its corresponding vector field $\phi(t)$</td>
<td>A-4</td>
</tr>
<tr>
<td>A-4.</td>
<td>Vector Field at a step or ramp edge point</td>
<td>A-5</td>
</tr>
<tr>
<td>A-5.</td>
<td>Vector field at a roof edge point</td>
<td>A-6</td>
</tr>
<tr>
<td>A-6.</td>
<td>Original of roof edge and edge points</td>
<td>A-7</td>
</tr>
<tr>
<td>A-7.</td>
<td>Clear edges and edges with noise added</td>
<td>A-10</td>
</tr>
<tr>
<td>A-8.</td>
<td>Roof edges with corresponding ROC curves</td>
<td>A-11</td>
</tr>
<tr>
<td>A-9.</td>
<td>Comparison of Sobel and Moment operator</td>
<td>A-12</td>
</tr>
<tr>
<td>A-10.</td>
<td>Signal to noise ratio vs. index of detectability</td>
<td>A-13</td>
</tr>
</tbody>
</table>
INTRODUCTION

This report contains a description of an experimental videotape reading system developed at the White Sands Missile Range Instrumentation Directorate computer lab for the investigation of image processing and pattern recognition concepts. The VRS is presently being used to study the concept of accurately determining the aspect angles of a target from one frame of video. The ability of accurately finding the position of a target from one frame of video is useful in extracting a data product from a videotape when there is tape available from only one station. Such a system, made into a real-time hardware machine, would also have applications in fire control of high-energy lasers, since the aiming of such devices requires that exact knowledge about the position of the target be available so that energy can be deposited at a critical point of the target. This experimental system is useful as a test bed for concepts that will have applications both in extracting a data product to be used by customers of WSMR and as a model for a hardware machine that would be used both for real-time tracking and for fire control of new weapons technology.

SYSTEM DESCRIPTION

The data flow of this system is as follows:

- A video tape of a mission is taken at a station.
- The frames to be read are put on a video disk which is attached to an image analyzing system capable of digitizing the frames in the video disc.
- These frames are digitized and put into data files with another file containing all the file names of the frames which are of interest.

The software then processes the data in the following sequence:

1. Read in file containing names of files to be processed.
2. Read in first file and do preprocessing on it until completely done.

begin
repeat
cobegin
begin
3a. Read next file and do preprocessing on it.
end;
begin
3b. Make a contour of the previously processed file and do the classification.
end;
end; until eof;
4. Finish off classifying last one read in
Processing begins by first doing pixel level operations. The classification is done by making a line drawing of the plane or rocket to be analyzed, and comparing it against a previously stored line drawing library made from views of the object in question at different angles. Before a contour of the target (Fig. la) can be made, points which are possible candidates for edge points must be identified. Since, typically, scenes that we process are very noisy, we begin by doing a three-by-three averaging to every point in the scene. After this, a moment edge detector is used to assign to each point in a scene a value which reflects the probability that a point is an edge point (Fig. lb). A threshold is chosen by the operator and all points classified as possible target points are assigned a zero and all others a one. The computer then makes and displays a contour of the entire scene with different polygonal segments being assigned different values (Fig. 2). The operator chooses the number of segments which make up the target, and the computer writes the segments out in a file. This file is then modified by the use of interactive graphics programs (Figs. 4 and 5). The result (Fig. 1c) is compared against the library of stored views (Fig. 3), the best match is found (Fig. 6) and the angular data needed is read from the coordinate system. A description of the operations that take place is thus:

begin
1. Read in file containing names of files to be processed.
2. Read in first file and do preprocessing on it until completely done.
repeat
cobegin
begin
3A. Read next file and do preprocessing on it.
end; begin
3B. Make a contour by the following process:
   a. Using a histogram, computer chooses a threshold for the moment file of the original and displays a contour based on this threshold.
   b. Is this contour acceptable?
   c. while contour not acceptable do
      begin
         *Obtain new threshold from operator.
         *Draw contour
         *Is contour acceptable?
      end
   d. Let operator choose segments that will be used to construct target.
   e. Display segments chosen by the operator and modify them as the operator instructs.
   f. Calculate the Fourier descriptors, normalize and do classification.
end; until eof;
4. Finish off classifying last one read in
end.
Figure 1a. Digitized video image of F102.

Figure 1b. Result of processing original with an edge detector.

Figure 1c. Contour of F102.
Figure 2. Line segments and associated segment numbers.
Figure 3. Library of contours of F102. This library was generated from a computer made three-dimensional model.
Figure 4. Some contours found by the computer before being corrected by the operator.
Figure 5. Contours of Figure 4 after being processed by the operator.
Figure 6. The best match found by the computer.
STRUCTURE OF THE PREPROCESSING SOFTWARE

Before a contour of the scene can be made, points which are possible candidates for edge points must be identified. Since, typically, the scenes that we process are very noisy we begin by doing a three-by-three, averaging to every point in the scene. After this a moment edge detector is used to assign to each point in a scene, a value which reflects the probability that a point is an edge point. As shown in Appendix A this sequence of steps increases the probability of detecting edge points. The next step is to do another averaging operation on the moment file with the purpose of increasing the connectedness of the edge points.

The software to accomplish the preprocessing was written with two ends in mind; one was that this software would be a model for a hardware module to be built later, and the other was that execution time be reduced by overlapping input/output with processing. Figure 7 illustrates how the software is set up. The programs READ1, WRITE1 and PROCESS are passive programs in that they suspend themselves immediately after doing some initialization operations. These consist of bookkeeping operations such as setting input file name, output file name, and setting up parameters so that the proper buffer is accessed each time a program is activated. The program which drives these passive programs is called MAIN1. It runs the needed programs and synchronizes them via the use of global event flags. After the preprocessing is finished it initiates the next step in processing by its call to ARROWS.

A typical frame is processed by MAIN1 in the following way: First the programs READ1, WRITE1 and PROCESS are loaded into memory. They do whatever initialization is necessary and then suspend themselves. There are two input buffers that will be used by READ1 to store the data to be processed, and two output buffers where processed data is put and from where the program WRITE1 writes the data out onto the disk. MAIN1 first has the two input buffers (lines 15 - 18) filled by the two activations of READ1 done by two calls to RESUME(READ1). READ1 automatically processes the buffers in an alternate manner as do PROCESS and WRITE1. The buffers are initially set up (lines 21 - 24) so that the remaining processing can be done concurrently (lines 25 - 35). In the do loop there are waits for flags to be set that indicate that each of the programs involved are finished. The rest of MAIN1 finishes up with the buffers that need to be processed and written out. On line 45 it starts the next step for this frame by its call to ARROWS.

```assembly
0015 CALL RESUME(READ1) ; FILL IN BUFFER #2
0016 CALL WAITFR(36)
0017 CALL CLREF(36)
0018 CALL RESUME(READ1)
0019 CALL WAITFR(36)
0020 CALL CLREF(36) ; BUFFERS #1 and #2 FULL
0021 CALL RESUME(PROCE1,)
0022 CALL WAITFR(37)
0023 CALL CLREF(37)
0024 CALL WAITFR(42)
```
Figure 7. Structure of the pre-processing software.
PEAD1 has access to a common global area (line 8) where it reads in the data to be processed. After initializing (lines 19 - 20) it suspends itself until activated. When activated it places the last six rows it has read in the first six positions of the buffer (lines 25 - 27) to be processed. It then reads 32 rows and stores the last six into STORE. The bookkeeping for the change of buffers is done (lines 32 - 34). MAIN is signalled that READ1 is finished by the call to SETEF and there is a jump to 21 which suspends READ1.
The next step after the files have been read is to do the averaging and edge detection. Again PROCE has access to the global common area DTA. It initializes itself and then suspends itself and waits for MAIN to activate it when needed. The processing (lines 13, 14, 15) consists of an averaging operation, an edge detection (MOMENT) and another averaging. The processing is done from INBUF to OUTBUF (AVG), OUTBUF to INBUF (MOMENT), and then INBUF to OUTBUF. The bookkeeping to allow alternate buffers to be processed is then done; the program suspends itself and then waits for the next call.
The average that is done is an unweighted average. The edge detector used is a moment operator which has been shown to perform well in the presence of noise. The next program that is called is WRITE1. The data structure here are the same as those used for READ1 with the same global common area being used. It also suspends itself and waits to be activated.
A model for a hardware realization of this software is given in Figure 8. Here each of the circles would be a CPU together with some local memory. They would be passive and controlled by a CPU, MAIN. The squares would correspond to buffers accessed by CPU's as indicated. There are standard hardware methods, such as interrupts and flags, that can be used for the synchronization which is done in the software model.
Figure 8. Generic hardware model for preprocessing hardware.
CONSTRUCTION OF A CONTOUR FROM THE MOMENT FILE

After the moment file \( \text{PIC} \) is created a threshold "T" must first be found such that a contour of the target will be included in the set.

\[
\{ \text{PIC}(i,j): i=1..n, j=1..N, \text{PIC}(i,j) < T \}
\]

To find \( T \) we first compute the histogram of the moment file created. A number \( P \) needs to have been chosen beforehand which represents the percentage of the scene points which are target points. The first point for which \( \sum_{X=0}^{255} \text{histo}_X > P \)

is found and used as the value of \( T \). It has been found that \( P = 15 \) works well in cases where the target is a small part of the scene and \( P = 25 \) does well when the target is a large percentage of the scene. In Figure 9 there are four originals that will be reduced to a contour. The result of preprocessing this data is in Figure 10. The problem now is to find a \( T \) such that the target will be separated from the background. If we look at the raw histograms (Figure 11I) we can, in some cases, guess at where the threshold should be chosen, assuming that there is one distribution for the target and another for the background. The background distribution is centered about the maximum of the histogram while the target distribution is part of the tail of the histogram. Thus it is reasonable to suppose that the target points constitute a certain percentage of the points to the right of some value. Experiments have shown that the proper value for this percentage is between 15 and 25, depending on the size of the target. Figure 11b is a figure found from 11a by graphing

\[
\sum_i (X) = \sum_{X=0}^{255} \text{histo}_X
\]

for each histogram of 11(I). From this graph we see that, as the contrast decreases, the threshold to be chosen decreases, a procedure that agrees with our intuition. We can also see that, when the target size is large, the graph is radically different than when the target is small. Using 11(II) and \( P = 23 \), we obtain the contours of Figure 12. The computer is set to threshold at \( P = 23 \), the contour appears on the screen; and the operator can reject this contour and request a new one based on an operator supplied value for \( P \). One choice of \( P \) does not always produce closed contours of the target; and this is why operator intervention is required at this point. As this system stands now, \( P \) is set by the operator on the initial frame and used for subsequent frames until the operator intervenes.
Figure 9a. F102 flying by a mountain.

Figure 9b. F102 flying in front of a mountain.

Figure 9c. Hawk missile

Figure 9d. F102, sunspots on wings and nose.
Figure 10a. Moment file of F102 flying by a mountain.

Figure 10b. Moment file of F102 flying in front of a mountain.

Figure 10c. Moment file of Hawk missile.

Figure 10d. Moment file of F102, sunspots on wings and nose.
Figure 11. Raw histogram Data (I) from moment file. Integrated Data (II) from raw histograms.
Figure 12a. Contour found from Moment file (Fig. 10a.)

Figure 12b. Contour found from Moment file (Fig. 10b.)

Figure 12c. Contour found from Moment file (Fig. 10c.)

Figure 12d. Contour found from Moment file (Fig. 10d.)
Once the value of $T$ has been determined so that the target lies in the set
\[
\{ \text{PIC}(i,j): i=1\ldots n, j=1\ldots m, \text{PIC}(i,j) < T \}
\]

We are in a position to begin making the contours from which the targets will be extracted. Again since these programs are intended to be used as models for a future hardware implementation the processing follows a sequential order. That is, the rows are processed from 1 to $n$ with the original gray values used just once in the processing. It is commonly known that to construct a fast contour plotting algorithm the Freeman code must be dropped as soon as possible. We use the Freeman code to record the direction as we go along but convert to a polygonal representation as soon as it is possible, as when there occur two consecutive Freeman codes with the same direction. This is the first step in the contour-forming process and results in a set of line segments which are specified by their endpoints. The next step is joining the elements into contours.

The program that performs the operations described above is the program ARROW. First of all, six rows to be processed are read into AMMOUN(128,6). One pair of rows is processed at a time with the results of the processing being put into STOR(128,2) (lines 52 - 63). The assignment of directions begins by first, thresholding two rows of AMMOUN, with a point being assigned a zero if the average of a two-by-two neighborhood is greater than $t$ and a one if it is less than $t$ (lines 59 - 61). Beforehand STOR(128,1) is first overwritten by the last row processed which had been put in STOR(128,2) (lines 65 - 66).

```
0001 PROGRAM ARROW
0002 INTEGER*2 MAG(384),ANG(384),CENTER,THRESH,X
0003 INTEGER*2 U,V,POINT3,AO,A1,A2,A3
0004 INTEGER*2 POINT1,POINT2,VAR,SUM,SIGN
0005 INTEGER*2 H0,H1,H2,H3,STOR(128,2),AVERAG,THRESH
0006 LOGICAL*1 Y,ANS,INV,FLAG
0007 LOGICAL*1 AMMOUN(128,6),ANGLE(128,6),NEW(128,5)
0008 LOGICAL*1 B1O(25),NAMETE(26)
0009 REAL*4 X1,SEGMN
0010 INTEGER*2 LINES(500,3),COORDI(1500,6)
0011 INTEGER*2 POINSI
0012 INTEGER*2 POINTL,POINTH,POINTE,COL
0013 COMMON INV,FLAG,SIGN
0014 COMMON LINES,POINTE,POINTH,POINTL,COL,128
0015 COMMON /DTA/COORDI
0016 EQUIVALENCE (MAG,AMMOUN)
0017 EQUIVALENCE (NAMETE,COORDI)
0018 EQUIVALENCE (ANG,ANGLE)
0019 DATA Y/89/
0020 DATA SEGMN/6RSEGMENT/
0021 DATA IJ/0/
0022 DATA AK/1.4111764/
0023 CALL ERSET(37,.TRUE.,.FALSE.,.FALSE.,.FALSE.,31)
0024 I=NAMETE(1)
0025 DO 999 M=1,25
0026 BI0(M) = NAMETE(M+1)
```
CONTINUE
CALL ASSIGN (7, B10, I)
DO 947 L = 1, 20
COORDI (L, 1) = 0
COORDI (L, 2) = 0
COORDI (L, 3) = 0
COORDI (L, 4) = 0
COORDI (L, 5) = 0
COORDI (L, 6) = 0
CONTINUE
POINSI = 1
DEFINE FILE 7 (0, 64, U, POINT2)
POINT2 = 1
CALL DEVIAT (THRESH, POINT2)
IJ = 0
POINTT = 1
POINL = 0
POINH = 1
CONTINUE
DO 150 K00 = 1, 25
DO 140 IO = 0, 5
READ (7, POINT2, END = 77) (MAG (I), I = 1 + K0 * 64, 64 + K0 * 64)
140 CONTINUE
CONTINUE
DO 10 J = 1, 5
DO 5 I = 1, 127
H2 = HO
H3 = HI
LCONS = I + 1
HO = AMMOUN (LCONS, J) .AND. 255
HL = AMMOUN (LCONS, J + 1) .AND. 255
AVERAG = (HO + HL + H2 + H3) / 4
STOR (I, 2) = 0
IF (AVERAG .LT. THRESH) STOR (I, 2) = 1
CONTINUE
CONTINUE
CALL OUT (STOR, J + IJ)
DO 756 IND = 1, 128
STOR (IND, 1) = STOR (IND, 2)
CONTINUE
CONTINUE
CONTINUE
CONTINUE
POINT2 = POINT2 - 1
IJ = IJ + 5
CONTINUE
CONTINUE
CALL CLOSE (7)
COORDI (1, 6) = POINTH
CALL CHANGE
CALL REQUES (SEGNUM)
CONTINUE
END
When the Subroutine OUT is called it uses STORE to generate Freeman-Code directions for a row and three such rows are stored in the array NUMBER(128,3).

To obtain a Freeman-Code, the patterns of Figure 13 are assigned the indicated values by the subroutine OUT and stored in NUMBER(128,3). All other patterns are assigned a -1.

```
0001 SUBROUTINE OUT(STORE,J)
0002 INTEGER*2 STORE(128,2),NUMBER(128,3)
0003 REAL*4 ANGLES(15)
0004 REAL*4 THETA
0005 LOGICAL*1 FLAG1,FLAG2,FLAG
0006 DATA ANGLES/-1.,7.,6.,3.,4.,-1.,5.,-1.,0.,-1.,2.,3*-1./
0007 DATA FLAG/.FALSE./
0008 IF(J.GT.3)CALL LOGIC(NUMBER,J)
0009 CALL READEF(15,LCODE)
0010 IF(LCODE.EQ.2) RETURN
0011 DO 10 I=1,127
0012 NUMBER(I,1)=NUMBER(I,2)
0013 NUMBER(I,2)=NUMBER(I+1,1)*2+ NUMBER(I+1,2)
0014 10 CONTINUE
0015 DO 20 I=1,127
0016 INDEX=STORE(I,1)*2**3+STORE(I,2)*2**2+STORE(I+1,1)*2+ STORE(I+1,2)
0017 IF(INDEX.EQ.0) INDEX=15
0018 NUMBER(I,3)=IINT(ANGLES(INDEX))
0019 20 CONTINUE
0020 RETURN
0021 END
```

The next step is the linking of directions which are the same, and appear sequentially in a three-by-three window. Two predicates are used in controlling the statements to be executed. These are

\[ P = \text{The element of NUMBER is a continuation of a segment of the same direction.} \]

\[ Q = \text{The element of NUMBER being checked is continued by a segment of the same direction.} \]

The cases where \( P = \text{true}. \) and \( Q = \text{true}. \) are illustrated in Figures 14a, b. The possible predicates and the actions taken when the predicates are true are shown in Figure 14c.

a. \( P \land Q \)

Put the segment number of the line that the element of NUMBER continues into first 15 bits of NUMBER(1,2). This segment number is extracted from the first 15 bits that NUMBER(1,2) continues.

b. \( \overline{P} \land Q \)

In this case a new segment needs to be started. The starting row and columns are stored in the array LINES. POINTL contains the current
Figure 13. Two-dimensional number patterns and their assigned geometrical directions.
segment number and \text{LINES}(\text{POINTL}, 1) \text{ sets the row number, } \text{LINES}(\text{POINTL}, 2) \text{ the column number and } \text{LINES}(\text{POINTL}, 3) \text{ the direction.}

c. \text{P and } \overline{\text{Q}}

This happens when a line segment of the same direction is terminated. The action taken in this case is to store the beginning coordinates ending coordinates, and direction of the line segment in \text{COORDI}. The beginning point of the segment is obtained by first stripping the first 15 bits off \text{NUMBER}(i, 2) and accessing the entry of \text{LINES} which corresponds to this number. This gives the starting point of the segment, while the final point is gotten from the current row and column coordinates.

d. \overline{\text{P}} \text{ and } \overline{\text{Q}}

It is an isolated direction and thus it is stored directly into \text{COORDI}.

Subroutines \text{LOGIC} and \text{GUARDS} look at the direction numbers in \text{NUMBER} and link those arrows that occur sequentially in the same direction. The arrays used to do the bookkeeping at this stage are \text{LINES}(,) and \text{COORDI}(,) with the final results being stored in \text{COORDI}(,). Long polygonal segments are constructed by tracking along consistent joins of these line segments at each point, checking for possible continuation of each segment.

The subroutine \text{LOGIC} drives the programs which produce the pseudo Freeman code and do the bookkeeping functions. The data from which it computes line segments is in \text{NUMBER}(128, 3) and it consists of the Freeman codes generated from the last three rows processed. The first fifteen bits of \text{NUMBER} are used to store the segment number of a particular entry.

\text{LOGIC} processes a row in the do loops of line fourteen to twenty-four. This loop begins by looking to see if the element \text{NUMBER}(128, 2) is a possible edge element, and if it finds that the element equals "-1" it looks at a new element since the non-edge elements have been assigned a "-1." If it is a possible edge element it extracts the segment number and calls \text{GUARDS} to compute the values of \text{P} and \text{Q}. What is left to do now are the actions which correspond to different values of \text{P} and \text{Q} this is done in lines 20 through 23.
LOGIC computes the values of $P$ and $Q$ by using the subroutine GUARDS. GUARDS looks in a three-by-three neighborhood of \texttt{STATES(I28,3)} and computes the value of $P$ and $Q$ for \texttt{STATES(I,2)}. It checks, as in Figure 14, for the appropriate values of $P$, lines 12 - 57, and then goes on to compute $Q$, line 60 - end. This processing is done for the entire file with the final results being stored into \texttt{COORDI}. \texttt{COORDI(1500,6)} is now sorted on the row coordinates of its elements. The format of the elements, that are stored in \texttt{COORDI}, is also changed so that the data is now

\begin{verbatim}
CODE,BEGIN(j),BEGIN(i),DEL
\end{verbatim}
Figure 14. Examples of geometric relations for various predicates, the geometric situation when (a) $P = \text{true}$, and (b) $Q = \text{true}$.
0028 STATES(I,2)=SEGMENT.OR.CODE
0029 GOTO 85
0030 IF((STATES(I-1,1).AND.7).EQ.3)SETMEN=STATES(I-1,1).AND."177770
0031 IF(SEGMEN.EQ.0) GOTO 85
0032 P=.TRUE.
0033 STATES(I,2)=SEGMENT.OR.CODE
0034 GOTO 85
0035 IF((STATES(I-1,2).AND.7).EQ.4)SEGMEN=STATES(I-1,2).AND."177770
0036 IF(SEGMEN.EQ.0) GOTO 85
0037 P=.TRUE.
0038 STATES(I,2)=SEGMENT.OR.CODE
0039 GOTO 85
0040 IF((STATES(I+1,1).AND.7).EQ.5)SEGMEN=STATES(I+1,1).AND."177770
0041 IF(SEGMEN.EQ.0) GOTO 85
0042 P=.TRUE.
0043 STATES(I,2)=SEGMENT.OR.CODE
0044 GOTO 85
0045 IF((STATES(I,1).AND.7).EQ.6)CHOCQ=-STATES(I,1)
0046 IF(CHOICP.EQ.-1) GOTO 85
0047 SEGMEN=CHOICP.AND."177770
0048 P=.TRUE.
0049 STATES(I,2)=SEGMENT.OR.CODE
0050 GOTO 85
0051 IF((STATES(I-1,1).AND.7).EQ.7)CHOCQ=STATES(I-1,1)
0052 IF(CHOICP.EQ.-1) GOTO 85
0053 SEGMEN=CHOICP.AND."177770
0054 IF(SEGMEN.EQ.0) GOTO 85
0055 P=.TRUE.
0056 STATES(I,2)=SEGMENT.OR.CODE
0057 GOTO 85
0058 CHOCQ=-1
0059 GOTO(100, 200, 300, 400, 500, 600, 700, 800)INDEX
0060 IF((STATES(I+1,2).AND.7).EQ.0)CHOCQ=-STATES(I+1,2)
0061 IF(CHOICQ.EQ.-1) GOTO 85
0062 Q=.TRUE.
0063 GOTO 85
0064 GOTO 85
0065 IF((STATES(I-1,3).AND.7).EQ.1)CHOCQ=STATES(I-1,3)
0066 IF(CHOICQ.EQ.-1) GOTO 85
0067 Q=.TRUE.
0068 GOTO 85
0069 IF((STATES(I,3).AND.7).EQ.2)CHOCQ=STATES(I,3)
0070 IF(CHOICQ.EQ.-1) GOTO 85
0071 Q=.TRUE.
0072 GOTO 85
0073 IF((STATES(I+1,3).AND.7).EQ.3)CHOCQ=STATES(I+1,3)
0074 IF(CHOICQ.EQ.-1) GOTO 85
0075 Q=.TRUE.
0076 GOTO 85
0077 IF((STATES(I+1,2).AND.7).EQ.4)CHOCQ=STATES(I+1,2)
0078 IF(CHOICQ.EQ.-1) GOTO 85
0079 Q=.TRUE.

28
SUBROUTINE BEGIN(DIRECT,J)

INTEGER*2 LINES(500,3),COORDI(1500,6)
INTEGER*2 COL,I,J,DIRECT,POINTL,POINTH,POINTT
LOGICAL*1 INV,FLAG
INTEGER*2 SIGN

COMMON INV,FLAG,SIGN
COMMON LINES,POINTL,POINTH,POINTT,COL,128
COMMON /DTA/COORDI
POINTL=POINTL+1
lines(POINTL,1)=I28
lines(POINTL,2)=J
lines(POINTL,3)=DIRECT
RETURN
END

SUBROUTINE DUMP(J)

INTEGER*2 LINES(500,3),COORDI(1500,6)
INTEGER*2 I,J,DIRECT,POINTL,POINTH,POINTT
REAL*4 BEGINX,BEGINY,ENDX,ENDY
INTEGER*2 SIGN,COL
LOGICAL*1 INV,FLAG

COMMON INV,FLAG,SIGN
COMMON LINES,POINTL,POINTH,POINTT,COL,128
COMMON /DTA/COORDI
DIRECT=LINES(COL,3)
BEGINX=30+LINES(COL,1)*7
BEGINY=775-LINES(COL,2)*7
ENDX=30+(128)*7
ENDY=775-(J)*7
GOTO(100,200,300,400,500,600,700,800)
DIRECT+1
ENDX=ENDX+7
GOTO 1000
BEGINX=BEGINX+7
BEGINY=BEGINY+7
GOTO 1000
BEGINX=BEGINX+7
GOTO 1000
BEGINX=BEGINX+7
GOTO 1000
BEGINX=BEGINX+7
GOTO 1000
BEGINX=BEGINX+7
GOTO 1000
SUBROUTINE CHANGE

INTEGER*2 COORDI(1500,6) ! INPUT DATA
INTEGER*2 POINT : AMOUNT OF DATA
LOGICAL*1 .FLAG,.B10(25)
INTEGER*2 LINES(500,3),POINTH,POINTT
INTEGER*2 COL,128,SIGN
LOGICAL*1 .IV
COMMON INV,.FLAG,.SIGN
COMMON LINES,.POINTL,.POINTH,.POINTT,.COL,.SIGN
COMMON /DATA/.COORDI
FLAG=.TRUE.
CALL ERSET(37,.TRUE.,.FALSE.,.FALSE.,.FALSE.,.FALSE..11)
kontinue(0)
CALL CONTINUE
C COORDI FORMAT=CODE,TAGLJ),TAGL11).DELTA

DO 90 1=2.POINT
GOTO(10,20,20,20,30,40,40,40) COORDI(1,3)+1
FLAG=COORDI(1,2).GE.COORDI(1,5)
COORDI(1,3)=IABS(COORDI(1,2)-COORDI(1,5))
IF (.NOT.,.FLAG)COORDI(1,4)=COORDI(1,2)
IF (.NOT.,.FLAG)COORDI(1,5)=COORDI(1,2)
GOTO 86
FLAG=COORDI(1,1),GE.COORDI(1,4)
COORDI(1,6)=IABS(COORDI(1,1)-COORDI(1,4))
IF (.FLAG) COORDI(1,4)=COORDI(1,2)
IF (.FLAG) COORDI(1,5)=COORDI(1,2)
GOTO 86

STOP
FLAG=COORDI(1,5).GE.COORDI(1,2)

COORDI(I,6)=IABS(COORDI(I,2)-COORDI(I,5))

IF(.NOT.FLAG) COORDI(I,4)=COORDI(I,1)

IF(.NOT.FLAG) COORDI(I,5)=COORDI(I,2)

GOTO 88

FLAG=COORDI(I,4).GE.COORDI(I,1)

COORDI(I,6)=IABS(COORDI(I,1)-COORDI(I,4))

IF(FLAG) COORDI(I,4)=COORDI(I,1)

IF(FLAG) COORDI(I,5)=COORDI(I,2)

GOTO 88

CONTINUE

CALL SORT(POINTH,2)

DO 199 I=3,POINTH-3

WRITE(9,250) COORDI(I,1),COORDI(I,2),COORDI(I,3),COORDI(I,4)

250 FORMAT(2X I1,X,13,X,13,X,13)

199 CONTINUE

RETURN

END
0031       L=K+1
0032       DO 3 J=L,R
0033       IF(COORDI(J,KEY).GE.COORDI(J-1,KEY))GOTO 3
0034       X(1)=COORDI(J-1,1)
0035       X(2)=COORDI(J-1,2)
0036       X(3)=COORDI(J-1,3)
0037       X(4)=COORDI(J-1,4)
0038       COORDI(J-1,1)=COORDI(J,1)
0039       COORDI(J-1,2)=COORDI(J,2)
0040       COORDI(J-1,3)=COORDI(J,3)
0041       COORDI(J-1,4)=COORDI(J,4)
0042       COORDI(J,1)=X(1)
0043       COORDI(J,2)=X(2)
0044       COORDI(J,3)=X(3)
0045       COORDI(J,4)=X(4)
0046       K=J
0047       3  CONTINUE
0048       R=K-1
0049       IF(L.LE.R) GOTO 1
0050       END
0051       SUBROUTINE SINGLE(STATES,J,ROW)
0052       INTEGER*2 I,J,DIRECT
0053       INTEGER*2 LINES(500,3),COORDI(1500,6)
0054       LOGICAL*1 R,S,NR,N4S,FLAG, INV
0055       INTEGER*2 SIGN,ROW,COL,POINTTH,POINTT,DIR
0056       INTEGER*2 CODE, SEGME,CHOICP,CHOICQ,POINSI
0057       REAL*4 BEGINX,BEGINY,ENDX,ENDY
0058       INTEGER*2 STATES(128,3)
0059       COMMON INV,FLAG,SIGN
0060       COMMON LINES,POINTL,POINTTH,POINTT,COL,I28
0061       COMMON /DTA/COORDI
0062       CODE=STATES(I28,2).AND.7
0063       DIRECT=CODE
0064       BEGINX=30+(128)*7
0065       BEGINY=775-(J)*7
0066       ENDX=30+(138)*7
0067       ENDY=775-(J)*7
0068       GOTO(100,200,300,400,500,600,700,800) DIRECT+1
0069       100       ENDX=ENDX+7
0070       ENDY=ENDY+7
0071       GOTO 1000
0072       200       BEGINX=BEGINX+7
0073       BEGINY=BEGINY+7
0074       GOTO 1000
0075       300       BEGINX=BEGINX+7
0076       GOTO 1000
0077       400       BEGINX=BEGINX-7
0078       BEGINY=BEGINY+7
0079       GOTO 1000
0080       500       BEGINX=BEGINX-7
0081       GOTO 1000
0082       600       ENDX=ENDX-7
0083       ENDY=ENDY-7
0084       GOTO 1000
0085       32
The fifth coordinate will be used to place pointers that will give the next piece of a particular polygonal line, if there is one. At this point the scene has been reduced to a number of line segments of different length, each having one of eight possible directions. The next step is to link these by checking to see if there is a possible continuation of one segment by some other segment. Such a linking of segments is the function of the program SEGARR. To begin with, all elements of COORDI(*,5) are set equal to zero, after which a number of segments is built up in the following steps:

1. Look through COORDI(*,5), and if an entry is found equal to zero then proceed, or else stop.

2. Start a new segment by recording the location of the zero entry of COORDI(*,5) in SEGS.

3. Now look for an element in COORDI that satisfies linking criteria as given in Figure 16. If such an element is found, two different cases will be considered. Either it is a single element, or it is a segment (more than one element). The two alternative courses of action are:

   - 'Segment' (a) Link-up data structures, as in Figure 15a, which results in the graphic operations (Figure 15c).

   - 'Single' (b) Link-up data structures, as in Figure 15b, which results in the graphic operations (Figure 15d).

After these segments have been created the segment list is looked through, and if there is a consistent join of two segments whose distance apart is less than three units, then these are joined.

The program SEGARR links together the segments which are stored in the common area DTA. The number of elements in COORDI is passed via the sixth element of COORDI (lines 9 - 11). COORDI(*,5) will be used to store the pointers and they are all initialized to zero in lines 16 - 18. The line with label 85 is the beginning of the code which constructs the polygons from the long line segments. First, COORDI(*,5) is searched for a zero, i.e., a segment that
Figure 15. Data structures (a), (b) and operations with their corresponding (c), (d) geometric structures and geometric operations.
hasn't been used in the construction of a polygon. If it finds a zero, it jumps out of that loop and begins constructing the data structure which corresponds to a polygon (lines 20 - 23).

The pointer which keeps track of the numbers of polygons POINTS gets updated, and the index of the segment to be processed gets stored both in SEGS(POINTS,1) and SEGS(POINTS,2); also an "-1" gets stored in COORDI(I,5) to indicate the end of a polygon (lines 26 - 30). The call to WHERE computes the endpoint of the current segment being analyzed and stores it in the array SEARCH(3). SEARCH(1) containing the direction of the segment and SEARCH(2), SEARCH(3) the column and row coordinates (lines 33 - 34). To determine if there is a segment of COORDI that continues the segment at POINTV a call to FIND is made on line 34. This Subroutine returns an 0 in FINDP if there is no continuation. It points to the continuation of the segment being analyzed if one has been found that passes the test in FIND (Figure 16, Lines 19 - 86). If a continuation has been found and it has COORDI(FINDP,J) = 0 then it is one of the original long line segments and it is added to the list being constructed (lines 35 - 40). Alternatively it may be that there is a continuation of the element being tested but that this continuation is a segment. In this case the list which is being tested is added to the continuation list (Lines 42 - 46) with the corresponding list operations as in Figure 15. The resulting contour has many of the important segments in it but there are many gaps in the contours (Figure 17) which should be closed. One obvious method of closing these is to search through all the segments and join those that are less than a certain distance apart (Lines 52 - 77). This works fairly well, as Figure 18 shows.

```
0001 INTEGER*2 SEARCH(3) ! : CODE,HEAD(J),HEAD(I)
0002 INTEGER*2 COORDI(1500,6) ! : CODE,BEGIN(J),BEGIN(I),DEL,POINTER
0003 INTEGER*2 POINTH !: ALIAS FOR POINT
0004 INTEGER*2 SEGS(500,4) ! : BEGIN,END
0005 INTEGER*2 POINTV,POINTS,POINT,DI,DJ,FINDP
0006 INTEGER*2 MIN,INDEX,H,T,Y(3)
0007 LOGICAL *1 BIO(25),FLAG,INTERN
0008 COMMON /DISOO/ SEGS,POINTS
0009 COMMON /DTA/COORDI
0010 CALL ERRSET(37,.TRUE.,.FALSE.,.FALSE.,.FALSE.,31)
0011 POINT=-COORDI(1,6)
0012 ISAVE=1
0013 K1=0
0014 602 CONTINUE
0015 85 CONTINUE ! RETURN HERE TO BEGIN A SEGMENT
0016 DO 90 I=ISAVE+1,POINT+3 ! SET ALL POINTERS TO ZERO
0017 90 CONTINUE
0018 DO 80 I=ISAVE+1,POINT+1 ! SEARCH FOR UNUSED ONES
0019 80 CONTINUE
0020 IF(COORDI(I,5).EQ.0) GOTO 70
0021 70 EXIT
```

35
Figure 16. Geometric criteria for the continuation of a polygonal segment.
Figure 17. Polygon before closing by distance measure.
Figure 18. Polygon after closing by distance measure.
0024 70 CONTINUE: JUMPS OUT OF LOOP HERE
0025 POINTV=1
0026 IF(POINTV.GT.(POINT)) GOTO 20: FINISHED
0027 POINTS=POINTS+1: UPDATE SEGMENT POINTER
0028 SEGS(POINTS,1)=1: MARK BEGINNING OF SEGMENT
0029 SEGS(POINTS,2)=I
0030 COORDI(POINTV,5)=-1
0031 60 CONTINUE: RETURN POINT FOR SEGMENT CONSTRUCTION
0032 L=POINTV: INTERMEDIATE STORAGE LOCATION
0033 CALL WHERE(COORDI(L,1),COORDI(L,2),COORDI(L,3),COORDI(L,4),SEARCH)
0034 CALL FIND(SEARCH,FINDP,POINTV,NP): FINDP POINTS TO NEXT OR 0
0035 IF(FINDP.EQ.0) GOTO 30: CANNOT CONTINUE
0036 IF(COORDI(FINDP,5).NE.0) GOTO 50
0037 COORDI(POINTV,5)=FINDP
0038 SEGS(POINTS,2)=FINDP
0039 COORDI(FINDP,5)=-1
0040 POINTV=FINDP
0041 GOTO 60
0042 50 CONTINUE: MERGE LISTS
0043 COORDI(POINTV,5)=SEGS(NP,1)
0044 SEGS(NP,1)=SEGS(POINTS,1)
0045 POINTS=POINTS-1
0046 GOTO 85: BEGIN A NEW SEGMENT
0047 COORDI(POINTV,5)=-1
0048 IF(SEGS(POINTS,1).EQ.SEGS(POINTS,2)) COORDI(POINTV,5)=0
0049 IF(SEGS(POINTS,1).EQ.SEGS(POINTS,2)) POINTS=POINTS-1
0050 GOTO 85
0051 20 CONTINUE
0052 K1=K1+1
0053 MIN=5000
0054 H=SEGS(K1,2)
0055 IF(H.EQ.-1) GOTO 110
0056 CALL WHERE(COORDI(H,1),COORDI(H,2),COORDI(H,3),COORDI(H,4),Y)
0057 DO 150 J=1,POINTS
0058 T=SEGS(J,1)
0059 IF(T.EQ.-1) GOTO 150
0060 DJ=ABS(COORDI(T,3)-Y(3))
0061 DI=ABS(COORDI(T,2)-Y(2))
0062 IDIS=DJ+DI
0063 IF(IDIS.LT.MIN) INDEX=J
0064 IF(IDIS.LT.MIN) MIN=IDIS
0065 150 CONTINUE
0066 IF(MIN.GT.3) GOTO 110
0067 ICONST=SEGS(INDEX,2)
0068 IF(SEGS(INDEX,1).EQ.COORDI(ICONT,5)) GOTO 110
0069 COORDI(SEGS(K1,2),5)=SEGS(INDEX,1)
0070 SEGS(K1,2)=SEGS(INDEX,2)
0071 IF(K1.EQ.INDEX) GOTO 110
0072 SEGS(INDEX,1)=-1
0073 SEGS(INDEX,2)=-1
0074 K1=K1-1
0075 110 CONTINUE

39
IF (K1 .NE. POINTS) GOTO 20
CONTINUE
CALL INITT(160)
CALL DWINDO(-50.,1000.,10.,850.)
CALL CHRSIZ(3)
CALL COMPLE(LX)
CALL DISPLA
CALL CHRSIZ(3)
CALL FINITT(0,780)

TYPE *,'$LX',LX
C ACCEPT*,IJO
C IF(IJO,EQ.0) STOP
FORMAT(3X,6(I5,X))
LTIME=1
CONTINUE
TYPE *,'* ?'
READ(5,222)IJO
IF(IJO,EQ.0) STOP
IF(LTIME,EQ.1)CALL INTTT(160)
LTIME=TIME+1
CALL DWINDO(-50.,1000.,10.,850.)
CALL CHRSIZ(3)
CALL DISPLA(IJO)
CALL CHRSIZ(3)
CALL FINITT(0,780)
IF(IJO.NE.0) GOTO 670

FORMAT(I3)
CALL CHRSIZ(3)
DO 500 I=1,POINTS
C IF(SEGS(I,4).LT.20) GOTO 500
C IF(SEGS(I,1).EQ.-1) GOTO 500
IHEAD=SEGS(I,1)
L=-SEGS(I,2)
CALL WHERE(COORDI(L,1),COORDI(L,2),COORDI(L,3),COORDI(L,4),SEARCH)
ICYCLE=TABS(COORDI(IHEAD,3)-SEARCH(3))

SUBROUTINE FIND(ACTIVE,FINDP,POINTV,NP)
INTEGER*2 ACTIVE(3) : POINT FROM WHERE SEARCH IS MADE
INTEGER*2 POINT
INTEGER*2 TEST1,TEST2
INTEGER*2 FINDP : INDEX OF POINT FOUND OR ZERO
INTEGER*2 COORDI(1500,6) : DATA TO BE SEARCHED
INTEGER*2 SEGS(500,4) : SEGMEN POINTERS
INTEGER*2 DI,DJ,DEL,POINTV
INTEGER*2 NP : INDEX OF SEGS FOR HEAD OF MERGE
INTEGER*2 POINTS
COMMON /DTA/COORDI
COMMON /DISO/SEGS,POINTS
POINT=COORDI(1,6)
FINDP=0
JDIS=1
JLIM=0
J2=ACTIVE(2)
I2=ACTIVE(3)
DO 5 L=100,1,-1
IF((POINTV+L).LT.1) GOTO 4
IF((POINTV+L).GT.POINT) GOTO 4
JN=COORDI(POINTV+L,2)
IN=COORDI(POINTV+L,3)
TEST1=IN-12
IF(IABS(TEST1).GT.JDIS) GOTO 4
TEST2=JN-J2
IF(IABS(TEST2).GT.JDIS) GOTO 4
GOTO(10,20,30,40,50,60,70,80) ACTIVE(1)+1
IF(TEST1.LT.JLIM) GOTO 4
GOTO 90
IF((TEST1.LT.JLIM).AND.(TEST2.GT.JLIM)) GOTO 4
GOTO 90
IF(TEST2.GT.JLIM) GOTO 4
GOTO 90
IF((TEST1.GT.JLIM).AND.(TEST2.GT.JLIM)) GOTO 4
GOTO 90
IF(TEST1.GT.JLIM) GOTO 4
GOTO 90
IF((TEST1.GT.JLIM).AND.(TEST2.LT.JLIM)) GOTO 4
GOTO 90
IF(TEST2.LT.JLIM) GOTO 4
GOTO 90
IF((TEST1.LT.JLIM).AND.(TEST2.LT.JLIM)) GOTO 4
GOTO 90
GOTO 90
CONTINUE
FINDP=POINTV+L
IF(COORDI(POINTV+L+5).EQ.0) RETURN ! A SIMPLE CONSTRUCT
DO 1 J=1,POINTS-1 ! IS CANDIDATE THE HEAD OF A LIST
NP=J
IF(SEGS(J,1).EQ.FINDP.AND.(COORDI(SEGS(J,2),5).EQ.-1)) RETURN ! IT IS T
CONTINUE
FINDP=0
IF((POINTV-L).LT.1) GOTO 5
IF((POINTV-L).GT.POINT) GOTO 5
JN=COORDI(POINTV-L,2)
IN=COORDI(POINTV-L,3)
TEST=IN-12
IF(IABS(TEST1).GT.JDIS) GOTO 5
TEST2=JN-J2
IF(IABS(TEST2).GT.JDIS) GOTO 5
GOTO(100,200,300,400,500,600,700,800) ACTIVE(1)+1
IF(TEST1.LT.JLIM) GOTO 5
GOTO 900
IF((TEST1.LT.JLIM).AND.(TEST2.GT.JLIM)) GOTO 5
GOTO 900
IF(TEST2.GT.JLIM) GOTO 5
GOTO 900
IF((TEST1.GT.JLIM).AND.(TEST2.GT.JLIM)) GOTO 5
GOTO 900
IF((TEST1.GT.JLIM).AND.(TEST2.LT.JLIM)) GOTO 4
GOTO 90
CONTINUE
This process results, in most cases, in a closed curve that can be analyzed by the Fourier Classification process but there are also cases where the contours produced are not suitable for processing but must be first modified by an operator (Figure 4, 5) before they can be used. In this case, the next step in the processing allows an operator to interactively modify the contours produced so that they are closed curves and can be analyzed by the Fourier descriptor programs. Programs that are used to modify the computer-generated polygons are documented in Reference 2. Examples of how they work are in Figures 3, 4, 19 and 20.

FOURIER DESCRIPTOR METHODS

The library that is stored in the computer does not use the \((x,y)\) coordinates of the polygons, but first transforms them via the Fast Fourier Transforms and stores the "Fourier Descriptors." These Fourier Descriptors are defined as follows: A closed curve can be thought of as a function of a complex variable, \(z(t)\), parametrized by arc-length \(t\). We can normalize and have the curve described by \(z(t)\), \(0 < t < 2\pi\). If we go around the contour more than once, we get a periodic function, which can be expanded in a convergent Fourier series. The Fourier Descriptor of the curve is defined to be the Complex Fourier series expansion of \(z(t)\) which is given by the formula

\[
Z(t) = \sum_{n=-\infty}^{\infty} A(n) e^{int}
\]

where

\[
A(n) = \frac{1}{2\pi} \int_{0}^{2\pi} z(t) e^{-int} dt
\]

(See Figure 21, 22)

Thus the Fourier Descriptors (32 of the \(A(n)\)) for each element in the library are computed and stored into memory. The contour of the unknown plane is then found, the Fourier coefficients for this unknown are calculated, and the angular data necessary is obtained by finding the element \(i\) the library whose Fourier coefficients are closest to the unknowns.

42
Figure 19. Polygons and corrections entered by the operator.
Figure 20. New polygons obtained from corrected polygons of Figure 19.
Figure 21. Three different contours of an F102 from video taken at WSMR. The Fourier coefficients are given in the format: absolute value, phase. The order is $A(1), A(2), \ldots, A(16), A(-15), \ldots, A(-2), A(-1)$. 
Figure 22. Three basic geometrical shapes and their Fourier coefficients.
In order to compute the Fourier Transforms, a closed curve description of the target to be analyzed has been produced by the computer and the operator. The description of this curve consists of a sequence of \(x,y\) coordinates, which are the vertices of a polygon. As the first step the length of this contour is computed, and the contour is resampled at a spacing chosen to make the total number of samples a power of two. This polygon is then filtered to remove noise, and the Fourier descriptor is computed by taking the Fast Fourier Transform of this sequence of \((x,y)\) coordinates.

If two polygons are congruent in the plane then they can be shown to be so by a sequence of rotations, translations and contractions followed by a point-to-point comparison. If we have two congruent triangles represented by a sequence of \(x,y\) coordinates they can be shown to be congruent by first rotating both so that their longest side lies on the \(x\) axis, doing separate contractions so that they both have the same area then doing a point-by-point comparison starting at the greatest \(x\) coordinate. It is clear that the point-by-point comparison must be done starting at the same place on both triangles, and continuing at equidistantly sampled points in order for this process to be meaningful. The geometric transformations used to show two polygons congruent translate into the frequency domain as shown in Table I.

**TABLE I. EQUIVALENT OPERATIONS**

<table>
<thead>
<tr>
<th>TIME DOMAIN</th>
<th>FREQUENCY DOMAIN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Translation</td>
<td>Addition to (a(0))</td>
</tr>
<tr>
<td>Rotation</td>
<td>Multiplication of series by a constant</td>
</tr>
<tr>
<td>Comparison point change</td>
<td>Multiplication of (a(j)) by (\exp(ijt))</td>
</tr>
</tbody>
</table>

In order for a comparison to be meaningful in the frequency domain, a "normalization" in the frequency domain must be done similar to the geometric normalization that has been done for the triangles. This normalization must be done, using only the operations which are listed on the right of Table I.

First, \(a(0)\) is set equal to zero to normalize position. Size normalization is accomplished by dividing each coefficient by the absolute value of \(a(1)\). To normalize the point where the comparison is to begin, we require that the phase of the two coefficients of largest magnitude be zero. For a simple closed curve that does not cut itself \(a(1)\) is the coefficient of largest magnitude. Some polygons and their normalized Fourier coefficients appear in Figure 22.

Let \(a(L)\) and \(a(K)\) be two non-zero coefficients of the Fourier series. The normalization multiplicity of the coefficients \(a(L)\) and \(a(K)\) is defined to be \(M=\text{abs}(K-L)\). Some of the geometric significance of \(M\) in the case where \(a(1)\) and \(a(L)\) are the only non-zero coefficients is given by the following proposition:

Let \(z(t) = A(1)\*\exp(it) + A(L)\*\exp(iLt)\) with \(\text{abs}(A(1)) > \text{abs}(A(L)) > 0\)

1. T. P. Wallace and O. R. Mitchell, "Local and Global Shape Description of Two and Three Dimensional Objects," School of Electrical Engineering, Purdue University, September 1979.
PROPOSITION 1. If \( \text{abs}(A(l)) > \text{abs}(L^*A(L)) \) then the closed curve described by \( z(t) \) has no intersections.

PROPOSITION 2. If \( \text{abs}(A(l)) = \text{abs}(L^*A(L)) \) then the closed curve described by \( z(t) \) has \( M = \text{abs}(l - L) \) cusp points. The angles at these points are convex if \( L < 0 \) and concave if \( L > 0 \) (see Figure 23).

PROPOSITION 3. The function \( \text{abs}(z(t)) \) has \( M \) maximum points and \( M \) minimum points.

Let \( z(t) = \sum_{j=-\infty}^{j=\infty} A(j) \exp(ijt) \)

PROPOSITION 4. The requirement that \( a(L) \) and \( a(K) \) have zero-phase angle can be satisfied by \( m \) different orientation/starting point combinations.

PROPOSITION 5. \( \max_j |A(j)| = \sum_{j=-\infty}^{j=\infty} |A(j)| \) if the associated curve has no intersections. 

Thus for a figure whose second greatest coefficient is \( a(L) \), there are \( M = \text{abs}(L - 1) \) possible ways to normalize this figure. In order for an accurate comparison to be possible the normalization chosen for like figures must be the same. We use the following method to choose the normalization:

1. Calculate the Fourier coefficients for the \( M \) possible normalizations.

2. For each of the \( M \) normalizations calculate

\[ \sum \text{re}(a_i) |a_i| \]

3. Use the normalization which maximizes the above quantity.

The pattern recognition method begins by constructing a three-dimensional representation of the target to be analyzed. A library of polygons is then constructed which are the projections of the three-dimensional object, as seen from different views. From this library of polygons a library of Fourier Descriptors is computed using normalization described above, and stored into the computer. When an unknown is to be analyzed and its contour is found, the Normalized Fourier Descriptors are calculated and these numbers are compared to the library entries via the difference.

\[ \sum |a_i - \text{LIB}_{j}(i)| \]

Figure 23. Contours generated by functions of the type $Z(t) = \exp(it) + \frac{1}{L} \exp(iLt)$ for (a) $L = 15$ (b) $L = 3$ (c) $L = -3$ (d) $L = -15$
Once the closest element of the library is found, an interpolation\textsuperscript{3} is done to get an accurate measure of the aspect angles. When this is done the processing of the frame is finished.

We have described an interactive system that could be used to obtain aspect angle information from one frame of video. Before this system can be made completely automatic there must be research done with regards to two difficult problems. One is the automatic choosing of a threshold which would separate possible target points from the background. An approach being considered is an adaptive procedure for choosing $P$ where $P$ would be incremented if the size of the ellipse defined by the coefficients $a(1)$ and $a(-1)$ increased; and $P$ would be decreased if the area of this ellipse decreased. The other is the extraction of the target from the polygonal representation of the scene. Both are difficult problems which will require much research before a satisfactory solution can be found.

---

\textsuperscript{3} T. P. Wallace and O. R. Mitchel, \textit{ibid.}
Appendix A

FINDING EDGES IN NOISY SCENES

Research into methods of identifying edges in a noisy scene has been an active field of investigation for many years. Treatment of the subject may be found in many books written over the past decade and many different approaches are proposed. Recently a survey and comparative analysis of the methods was made.

The body of this appendix is segmented into four parts. In the first, we derive and define a "Moment Operator" which we show to work well for step and ramp edges. Then, we define and characterize second order edges using the concept of the rotation of a point in a vector field and develop the detector analytically. In Section 3 we develop the algorithms for implementing the previously defined operators. Finally, in Section 4, these algorithms are evaluated using ROC curves and compared with previously known techniques.

The detection of edges to isolate objects in a scene is motivated by many distinct problems. One such problem arises in a tracking system where the input video image is analyzed and the object to be tracked identified. Subsequent input and feedback to the drive controls causes the sensor to re-orient to a new position in an attempt to maintain the same x-y coordinate position for the object in the field of view. While this problem motivated the research that led to this paper, the results herein discussed are much broader in scope and application. The constraints imposed by this problem led to a method that is useful in high data throughput systems.

Section 1. Edges from Moments

First order edge detection methods work in the following way: A picture function $f(x,y)$ is transformed to another picture function $F(x,y) = T f(x,y)$ in such a way that the edges of objects in the scene will be in the set $\{(x,y): F(x,y) > W\}$ for some $W$. The usual method is to transform the picture using $T$ equal to the gradient operator. Different edge detection methods correspond to different numerical approximations to the gradient.

7. I. Abdou, "Quantitative Methods of Edge Detection," Image Processing Institute, University of Southern California, Los Angeles, California, 1978
The method used in our edge detection program is not based on derivatives. To reduce the effect of noise, this edge detection method uses integrals.

The reasoning for the use of moments to find edges is as follows:

- A digitized picture can be thought as a lamina whose density at each point is \( f(x,y) \), so points of high intensity correspond to points of high density.

- A point \((a,b)\) on an edge in the original function (see Figure A-1) would correspond to a point in this lamina (digitized picture) with high densities on one side and lower densities on the other side.

Thus if we look at a small lamina centered at point \((a,b)\) and compute the center of mass of this small lamina, we can expect the center of mass to lie within an area of high densities.

\[
\begin{align*}
\text{Figure A-1. Example center of mass vectors for (1) and edge and (2) a region of uniform intensity.}
\end{align*}
\]

Suppose we now look at a point \((c,d)\) such that the densities around it are fairly constant. Then the center of mass of a small lamina about it would be close to \((c,d)\). In this case, a vector from \((c,d)\) to the center of mass would be very small compared to a vector from \((a,b)\) to the center of mass in the previous case.

We conclude that one way to transform \( f(x,y) \) to \( F(x,y) \) such that edges of the original picture lie in the set \( F(x,y) > W \) is to replace every \( f(x,y) \) by the length of the vector from \((x,y)\) to the center of mass of a small lamina centered about \((x,y)\). That is, \( F(x,y) \) is the magnitude of the vector from \((x,y)\) to the center of gravity of a square lamina centered at \((x,y)\) whose density is given by the picture function \( f(x,y) \).

Figure A-2(b) is an example of how this method works on a scene (Figure A-2(a)) typical of those we study at WSMR.
(a) Image of rocket and plume. The plume is the large region of highest intensity.

(b) Ramp and step edges found by using the moment operator.

(c) The vector field generated by the moment operator.

(d) Second order edges detected by using the vector field.

Figure A-2. Rocket and results of processing.
Once the coordinates \((X,Y)\) of the center of mass of a lamina about \((x,y)\) are calculated, the direction of the edge (if any) can easily be found. Since \((X,Y)\) points to where the intensity of the picture is the highest, the direction of the edge is perpendicular to the direction of the vector from \((x,y)\) to \(X,Y\). If we take \((x,y) = (0,0)\), then the direction of the edge is \(\theta = \arctan (Y/X) + \pi/2\).

Thus this model gives for each point in the scene a quantity that measures the probability that a point is an edge point and a direction which is the direction of a possible edge through that point.

The model introduced in Section I will not work for roof edges. This is because at the very peak of the roof, exactly where the edge is situated, both \(X\) and \(Y\) are equal to zero. In order to detect roof edges we need to take advantage of the direction information, and as Figures 6(a), (b) and (c) show we need to detect the shearing cause by the change in direction of the vector field at the edge points. One way of doing this is by using a tool from the theory of vector fields, namely the rotation of a vector field about a point.

If a curve \(\Gamma\) on the plane (scene) is given in the form
\[
\Gamma: x = X(t), \ y = y(t) \quad a \leq t \leq b
\]
then \(\phi(t) = (\psi[x(t), y(t)], \psi[x(t), y(t)]\) is defined on the interval \([a,b]\) (see Figure A-3).

![Figure A-3. A curve \(\Gamma\) and its corresponding vector field \(\phi(t)\)](image)

For each \(t \in [a,b]\) there is determined an angle, the angle in radians between \(\phi(t)\) and \(\phi(a)\) measured from \(\phi(a)\) to \(\phi(t)\). This angle is a many-valued function (vanishing for \(t = a\)) is designated by \(\Phi(t)\) and called an angular function of the field \(\phi\) on a curve \(\Gamma\). The rotation of the field \(\phi\) on the curve \(\Gamma\) is defined to be
\[
\gamma(\phi, \Gamma) = \frac{1}{2\pi} \left[ \Phi(b) - \Phi(a) \right]
\]
If \( \Gamma \) is a closed Jordan curve, then the rotation is found by subdividing \( \Gamma \) into two curves (not closed), computing the rotation of each, and adding. In the following, \( \Gamma \) is taken to be a small circle about a point.

We can write the rotation as

\[
\gamma = \frac{1}{2\pi} [\theta(b) - \theta(a)] = \frac{1}{2\pi} \int_0^1 \frac{d\theta(t)}{dt} dt.
\]

With \( \theta(t) = \arctan \frac{Y}{X} + \pi/2 \), we make the following observations:

If \( \theta(t) \) is constant, then \( \frac{d\theta(t)}{dt} = 0 \) and \( \gamma = 0 \). So \( \gamma = 0 \) when \( x \) = a point on the edge of an object in a scene (see Figure A-4).

![Figure A-4. Vector Field at a step or ramp edge point.](image)

Section 2. Second Order Edges

After a scene is processed by the moment edge detector, each point is assigned a direction and a magnitude. In effect this specifies a vector at each point of the plan in question; i.e., these vectors define a vector field over the scene. An important tool in the study of vector fields is the rotation of a vector field.\(^8\),\(^9\) To define the rotation of a vector field, suppose a vector of the vector field \( \theta \) at the point \((x,y)\) is given by

\[
\phi(x,y) = (\phi(x,y), \psi(x,y))
\]

\[
\phi(x,y) = X(x,y)
\]

\[
\psi(x,y) = Y(x,y)
\]

If \( \theta \) is symmetric about \( x \) and \( \Gamma \) is a small circle about \( x \) = edge point on a roof edge, see Figure 5, then write \( \Gamma = \Gamma_1 + \Gamma_2 \) (where \( \Gamma_1 \) = one half of the circle and \( \Gamma_2 \) = the other half).


\[
\int_{\Gamma} \frac{d\Theta(t)}{dt} dt = \int_{\Gamma_1} d\Theta(t) + \int_{\Gamma_2} d\Theta(t) = \pi + \pi = 2\pi
\]

Figure A-5. Vector field at a roof edge point.

Figure A-6(a) and A-6(b) are examples of how these observations can be used to detect second order edges.

Section 3. Algorithms for Implementation

a. Calculation of Moment. Since we are interested in real time applications of these methods we simplify the calculation of \( \overline{X} \) and \( \overline{Y} \) by setting

\[
M = \int_{-h}^{h} \int_{-k}^{k} f(x + t, y + u) dt du = 1
\]

This can be justified by observing that \( M/4hk \) is the average of the intensities over a small neighborhood of \((x,y)\) and so this value can be approximated by the average value of intensities over the entire picture. This would then be just a scale factor and so could be left out.

To calculate the integrals involved we use an integral formula of order \( O(h^6) \). The formula for integration is

\[
\int \int F(x,y) = \sum_{i=1}^{9} W_i \cdot D_i \text{ with } W_{2k} + 1 = 25/324, W_{2k} = 10/81
\]

and if we apply this to the integrals for \( \overline{X} \) and \( \overline{Y} \) and factor out all scale factors we get

\[
\overline{Y} = 5 \ast (D_1 - D_5) + 4 \ast (D_8 + D_2 - D_6 - D_4)
\]

\[
\overline{X} = 5 \ast (D_7 - D_3) + 4 \ast (D_8 + D_6 - D_2 - D_4)
\]

and use \( \text{abs} (X)^2 - \text{abs} (Y)^2 \) for the associated magnitude. If we sweep a three-by-three window across digitized scene \( D_7 \) can be taken as the upper

---

Figure A-6. Original of roof edge and edge points.
left hand corner while D3 is the lower right hand corner. In this case the
direction of a possible edge is equal to
\[ \theta = \arctan \left( \frac{Y - X}{X - Y} + \frac{\pi}{2} \right) \]

b. Calculation of the Rotation. The vector field of a roof edge will
look like the vector field of Figure A-5. To find roof boundary points, we
therefore have to find points for which, in a neighborhood of such a point,
\[ \int_C \theta \, d\theta = 2\pi \]
The smallest region, in the discrete case over which we can take an integral,
is a two-by-two window; thus our algorithm sweeps a two-by-two window across
a scene and computes the integral \( \int \theta \, d\theta \) for each of these windows. If it
turns out that this integral is equal to \( 2\pi \), those four points which make up
the window are classified as boundary points. To calculate the integral of
a two-by-two window we use an approximation
\[ \int_\theta \sum_\ell \Delta \theta \]
\[ \ell = 1 \]
computed by a computer program.\(^11\)

For the purpose of this experiment the procedure used to generate a file
which is the file of detected second order edges in the following:

1. From the original file (scene) two files are generated; one (ACI)
   contains \( \sqrt{(X)^2 + (Y)^2} \); and the other (ANG) the angle of \( \theta \), \( 0 \leq \theta \leq 255 \)
a possible edge.

2. From the ANG and ACI files one new file AAA is created. AAA is
   created by sweeping a two-by-two window across the ANG file. The rotation
   is calculated, and if a point is classified as boundary, then to the corres-
   ponding point of AAA (initialized at zero) is added the average of those ele-
   ments of ACI that have the same subscripts as those of the two-by-two window
   being swept across ANG.

Examples of how this method works are shown in Figure A-6.

\[^11\] R. Machuca and A. Gilbert, "Finding Edges in Noisy Scenes, IEEE
Transactions on PAMI, unpublished.
Section 4. Evaluation

The methods described above were tested on disks whose edges were step, ramp and roof edges. The step and ramp edges had edge height equal to 16 while the roof edge was constructed by beginning at the center with gray value equal to 100 incrementing by one to gray value equal 132 and then decrementing by one to gray value 100. All files were 128 x 128 x 8.

To test the effectiveness of the different operations considered here we added Gaussian noise of different standard deviation to achieve a given signal to noise ratio and then tested the algorithms (Figure 7).

The SNR ratio was measured in db; that is, we used $\text{SNR} = 10 \log_{10}\left(\frac{16}{\sigma_n^2}\right)$ where $\sigma_n$ = standard deviation of the noise. For the ramp and step edges we used $\text{SNR} = 4, 5, 6, \ldots, 14$ while for the roof edge the signal to noise ratios used were 10, 11, 12, $\ldots$, 20. To measure the effectiveness of the different algorithms we graphed $\text{PF} = \text{the probability of false alarms vs. PD = the probability of detection}$ (Figure A-9). Figure A-8 contains examples of processed roof edge disks with $\text{SNR} = 13$. The graphs of $\text{PF vs. PD}$ (ROC curves) for the corresponding operators appears in Figure A-8.

The results for different operators and step, ramp and roof edges appear respectively in Figure A-10. These graphs show that the performance of the moment operator is, in all cases, better than that of the Sobel operator. A significant improvement is obtained by first applying the average and then the moment operator. When the signal-to-noise ratio is high the median gives better results than the average; but there is a cross-over point at which the average filter gives better results than the median.

---

Figure 1-7. Clear edges and edges with noise.
Figure A-8. Roof edges with corresponding ROC curves.
Figure A-9. Comparison of Sobel and moment operators.
Figure A-10. Signal to noise ratio vs. index of detectability.
REFERENCES

1. T. P. Wallace and O. R. Mitchell, "Local and Global Shape Description of Two and Three Dimensional Objects," School of Electrical Engineering, Purdue University, September 1979.


<table>
<thead>
<tr>
<th><strong>Organization</strong></th>
<th><strong>Number of Copies</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>STEWS-NR-A</td>
<td>1</td>
</tr>
<tr>
<td>CCNC-TWS</td>
<td>2</td>
</tr>
<tr>
<td>STEWS-NR-D</td>
<td>4</td>
</tr>
<tr>
<td>STEWS-PL</td>
<td>1</td>
</tr>
<tr>
<td>STEWS-PT-AL</td>
<td>3</td>
</tr>
<tr>
<td>STEWS-QA</td>
<td>1</td>
</tr>
<tr>
<td>STEWS-ID</td>
<td>1</td>
</tr>
<tr>
<td>STEWS-ID-D</td>
<td>1</td>
</tr>
<tr>
<td>STEWS-ID-O</td>
<td>1</td>
</tr>
<tr>
<td>STEWS-ID-E</td>
<td>1</td>
</tr>
<tr>
<td>STEWS-ID-P</td>
<td>3</td>
</tr>
<tr>
<td>STEWS-ID-T</td>
<td>1</td>
</tr>
<tr>
<td>STEWS-PT-AM</td>
<td>1</td>
</tr>
</tbody>
</table>

Commander
US Army Test and Evaluation Command
ATTN: DRSTE-AD-I
Aberdeen Proving Ground, Maryland 21005

Commander
Army Materiel Development and Readiness Command
ATTN: DRCAD-P
5001 Eisenhower Avenue
Alexandria, Virginia 22333

Director of Research and Development
Headquarters, US Air Force
Washington, DC 20315

Director
US Naval Research Laboratory
Department of the Navy
ATTN: Code 463
Washington, DC 20390
<table>
<thead>
<tr>
<th>No. of Copies</th>
<th>DISTRIBUTION LIST (cont)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Commander</td>
</tr>
<tr>
<td>1</td>
<td>Air Force Cambridge Research Center</td>
</tr>
<tr>
<td>1</td>
<td>Commander</td>
</tr>
<tr>
<td>1</td>
<td>US Naval Ordnance Test Station</td>
</tr>
<tr>
<td>2</td>
<td>Director</td>
</tr>
<tr>
<td>2</td>
<td>National Aeronautics and Space Administration</td>
</tr>
<tr>
<td>1</td>
<td>AFATL/DLODL</td>
</tr>
<tr>
<td>2</td>
<td>Commander</td>
</tr>
<tr>
<td>1</td>
<td>Pacific Missile Test Center</td>
</tr>
<tr>
<td>2</td>
<td>Commanding Officer</td>
</tr>
<tr>
<td>2</td>
<td>Naval Air Missile Test Center</td>
</tr>
<tr>
<td>3</td>
<td>Office of the Chief</td>
</tr>
<tr>
<td>2</td>
<td>Commanding Officer</td>
</tr>
<tr>
<td>1</td>
<td>US Army Ballistics Research Laboratories</td>
</tr>
<tr>
<td>1</td>
<td>Commanding Officer</td>
</tr>
<tr>
<td>1</td>
<td>US Army Electonic Command</td>
</tr>
<tr>
<td>1</td>
<td>Commander</td>
</tr>
<tr>
<td>1</td>
<td>US Army Research Office</td>
</tr>
</tbody>
</table>

| Commander | Air Force Cambridge Research Center |
| L. G. Hanscom Field | ATTN: AFCS |
| Bedford, Massachusetts 01731 |

| Commander | US Naval Ordnance Test Station |
| ATTN: Technical Library |
| China Lake, California 93555 |

| Director | National Aeronautics and Space Administration |
| ATTN: Technical Library |
| Goddard Space Flight Center |
| Greenbelt, Maryland 20771 |

| AFATL/DLODL | Eglin Air Force Base |
| Florida 32542 |

| Commander | Pacific Missile Test Center |
| Point Mugu, California 93041 |

| Commanding Officer | Naval Air Missile Test Center |
| Point Mugu, California 93041 |

| Office of the Chief | Research and Development |
| Department of the Army |
| Washington, DC 20310 |

| Commanding Officer | US Army Electonic Command |
| Meteorological Support Activity |
| ATTN: Technical Library |
| Fort Huachuca, Arizona 85613 |

| Commanding Officer | US Army Ballistics Research Laboratories |
| Aberdeen Proving Ground, Maryland 21005 |

<p>| Commanding Officer | US Army Research Office |
| P. O. Box 12211 |
| Research Triangle Park, North Carolina 27709 |</p>
<table>
<thead>
<tr>
<th>DISTRIBUTION LIST (cont)</th>
<th>No. of Copies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Commander</td>
<td>1</td>
</tr>
<tr>
<td>Atlantic Missile Range</td>
<td></td>
</tr>
<tr>
<td>Patrick Air Force Base, Florida 32925</td>
<td></td>
</tr>
<tr>
<td>Commanding Officer</td>
<td>1</td>
</tr>
<tr>
<td>US Army Aviation Test Activity</td>
<td></td>
</tr>
<tr>
<td>Edwards Air Force Base, California 93523</td>
<td></td>
</tr>
<tr>
<td>Defense Documentation Center</td>
<td>12</td>
</tr>
<tr>
<td>Cameron Station</td>
<td></td>
</tr>
<tr>
<td>Alexandria, Virginia 22314</td>
<td></td>
</tr>
<tr>
<td>US Army Materiel Systems Analysis Agency</td>
<td>1</td>
</tr>
<tr>
<td>ATTN: DRXSY-MP</td>
<td></td>
</tr>
<tr>
<td>Aberdeen Proving Ground, Maryland 21005</td>
<td></td>
</tr>
<tr>
<td>Director</td>
<td>1</td>
</tr>
<tr>
<td>US Naval Research Laboratory</td>
<td></td>
</tr>
<tr>
<td>Department of the Navy</td>
<td></td>
</tr>
<tr>
<td>ATTN: Dr. J. R. Slagel</td>
<td></td>
</tr>
<tr>
<td>Washington, DC 20378</td>
<td></td>
</tr>
<tr>
<td>Chief</td>
<td>1</td>
</tr>
<tr>
<td>Department of the Air Force</td>
<td></td>
</tr>
<tr>
<td>Mathematical Analysis Branch</td>
<td></td>
</tr>
<tr>
<td>ATTN: Dr. H. L. Oestreicher</td>
<td></td>
</tr>
<tr>
<td>Wright Patterson Air Force Base, Ohio 45433</td>
<td></td>
</tr>
<tr>
<td>Commander, Dept. of the Army</td>
<td>1</td>
</tr>
<tr>
<td>US Army Research Office</td>
<td></td>
</tr>
<tr>
<td>ATTN: Dr. Frank Kuhl, Bldg. 95N</td>
<td></td>
</tr>
<tr>
<td>P.O. Box 12211</td>
<td></td>
</tr>
<tr>
<td>Dover, New Jersey 07801</td>
<td></td>
</tr>
<tr>
<td>Commander, US Army Research Office</td>
<td>1</td>
</tr>
<tr>
<td>Department of the Army</td>
<td></td>
</tr>
<tr>
<td>ATTN: Dr. W. Sander</td>
<td></td>
</tr>
<tr>
<td>P. O. Box 12211</td>
<td></td>
</tr>
<tr>
<td>Research Triangle Park, North Carolina 27709</td>
<td></td>
</tr>
<tr>
<td>Commander, Dept. of the Army</td>
<td>1</td>
</tr>
<tr>
<td>Office of the Assistant Secretary</td>
<td></td>
</tr>
<tr>
<td>ATTN: Dr. J. H. Yang</td>
<td></td>
</tr>
<tr>
<td>Washington, D.C. 20310</td>
<td></td>
</tr>
<tr>
<td>DISTRIBUTION LIST (Cont)</td>
<td>No. of copies</td>
</tr>
<tr>
<td>--------------------------</td>
<td>--------------</td>
</tr>
</tbody>
</table>
| Commander, Dept. of the Army  
Headquarters, US Army, TECOM  
ATTN: Mr. Ben S. Goodwin  
Aberdeen Proving Ground, Maryland  
21005 | 1 |
| Commander  
Department of the Army  
Office of the Assistant Secretary  
ATTN: Dr. E Yore  
Washington, D. C. 20310 | 1 |
| Commander, Dept. of the Army  
U. S. Army Armament Research and Development Command  
ATTN: Dr. A. Gyoroc, EC & SCWSL  
Dover, New Jersey 07801 | 1 |