AD-A076 112   ARMY ENGINEER TOPOGRAPHIC LABS FORT BELVOIR VA   F/G 9/4
A WEIGHTED LINE-FINDING ALGORITHM.(U)
SEP 79 K ABDOSHAH , A KLINGER
UNCLASSIFIED   ETL-0200   NL

|OF|
AD
A076112

END
DATE
FILMED
12-79
DDC

ETL-0200

12

LEVEL II

A weighted line-finding algorithm

K. Abdoshah

A. Klinger

SEPTEMBER 1979

U.S. ARMY CORPS OF ENGINEERS
ENGINEER TOPOGRAPHIC LABORATORIES
FORT BELVOIR, VIRGINIA 22060

E T L

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER 14 ETL—0200	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) 6 A WEIGHTED LINE—FINDING ALGORITHM.		5. TYPE OF REPORT & PERIOD COVERED 9 Research Note
		6. PER~~FORMING ORG. REPORT~~ NUMBER
7. AUTHOR(s) 10 K. Abdoshah A. Klinger		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS U. S. Army Engineer Topographic Laboratories Fort Belvoir, VA 22060		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS R3205
11. CONTROLLING OFFICE NAME AND ADDRESS U. S. Army Engineer Topographic Laboratories Fort Belvoir, Virginia		12. REPORT DATE 11 Sept~~ember~~ 1979
		13. NUMBER OF PAGES 21
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) 12 24		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)

picture processing	edge detection	Hough transformation
imagery	colinear points,	digital cartography
pattern recognition	line--finding transformation	
line detection	weighted line--finding transformation	

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

This paper introduces two new algorithms to detect a line in a digitized picture. The algorithms are compared with the Hough algorithm, and their computational advantages are shown. We discuss the potential for, and the use of, the algorithms in applications, including the use of imagery in digital cartography.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

403 192 LB

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)

CONTENTS

ILLUSTRATIONS

TABLES

A WEIGHTED LINE-FINDING ALGORITHM

I. INTRODUCTION

This paper presents two new algorithms to detect a line in a digitized picture. We restrict ourselves here to the simplest case, a digitized picture with binary black and white pixels, although detecting colinear points in a general picture is possible with some modifications in the algorithms.

The ultimate use of such algorithms could be:

1. To take digitized imagery into a linearized reduced-data format that could be rapidly compared to an existing digital map
2. To speed up automatic line-following procedures that aid photo interpretors, and in general
3. To facilitate the process of using image data in mapping.

In [1] Duda and Hart described a method due to Hough ("Hough transform") that replaces finding colinear points by an equivalent problem of finding intersections in the sinusoidal curves they generate by that transform. That is, each pixel (X_i, Y_i) in the image space is transformed into an r-θ parameter space curve:

$$(1)\ r = X_i \cos(\theta) + Y_i \sin(\theta) \qquad -R = \leq r \leq = R \qquad 0 = \leq \theta \leq \pi$$

The parameter space curves that correspond to three or more colinear points, pass through a single point (r_i, θ_i), which corresponds to the line that connects the points.

Section 2 presents the new line-finding algorithm (LF). In LF taking each pair of image points defines a line and this line is transformed into a point (r_i, θ_i) in the parameter space. In this the algorithm differs from the Hough transform algorithm which operates on individual image points taking them into parameter points. The new algorithm has the major advantage of detecting perturbed lines (lines which are broken into several parts or have portions shifted slightly from the original position) as entities.

Section 3 presents an extension of the line-finding algorithm, called the weighted line-finding algorithm (WLF). By using WLF it is possible to define a specific criterion for a searched line, for example distinguishing connected lines from disconnected (dashed) ones, or locating a feature involving a jagged line.

II. LINE-FINDING ALGORITHM

In this algorithm each pair of image points defines a line and this line is transformed into a point (r_i, θ_i) in the parameter space. Here r_i is the distance to the origin from the line, and θ_i is the angle between the normal from the origin to the image line, and the x-axis.

1. Algorithm

Consider a line passing through a pair of points (X_1, Y_1) and (X_2, Y_2). It is shown in Appendix A that the distance of the origin from the line will be

$$(2)\quad |\,r\,| = \frac{|\,X_1 \cdot Y_2 - X_2 \cdot Y_1\,|}{\sqrt{(X_1 - X_2)^2 + (Y_1 - Y_2)^2}}$$

The main steps in the analysis based on fig. 2, are that the distance between the origin and the intersection of the line and the x-axis is

$$(3)\quad OA = (X_2 Y_1 - X_1 Y_2) \,/\, (Y_1 - Y_2)$$

and the resultant transformation parameters are:

$$(4)\quad \theta = \frac{\pi}{2} + \arctan\,[\,(Y_1 - Y_2) \,/\, (X_1 - X_2)\,]$$

$$(5)\quad r = \begin{matrix} +\,|\,r\,| & \text{if } OA < 0 \text{ or } \theta < \frac{\pi}{2} \\ -\,|\,r\,| & \text{otherwise} \end{matrix}$$

To implement this algorithm, the parameter space is quantized into cells and an accumulator is assigned to each cell in the two-dimensional array. This quantization can be confined to the region $0 \leq \theta \leq \pi$, $-R \leq r \leq R$. Points outside this parameter–space rectangle correspond to lines in the image that do not cross the retina. R is the radius of the retina, the smallest circle centered at the origin surrounding the image (see fig. 1).

Each pair of points in image [(X,Y)] space creates a corresponding point in parameter space counted by a corresponding accumulator increment. After processing all image pixels, the program inspects the array to locate cells with high counts. For picture region edges that digitized into straight lines on the scanning retina, the significant of a higher cell count is a larger number of colinear image points. The relation between a cell's count and the number of colinear points in the image is not a linear function. In fact, it acts so that those edges which are perturbed (broken into several parts by digitization) are smoothed by transformation and subsequent inverse transformation. This is the main advantage of LF transformation over the Hough transformation.

2. Example 1

To compare the Hough algorithm with LF algorithm, they were both implemented by the first author on the PDP/11 computer at the UCLA Computer Science Department, and applied to the data shown in fig. 3, a 20 x 20 picture with many colinear points.

Quantizing θ at k = 20 nine-degree increments in θ, and quantizing r into l = 28 one-unit increments in r, yields the two-dimensional accumulator array of Table 1 for the Hough algorithm, and Table 2 for LF algorithm.

Thresholds selected for the two techniques permitted detection of line #4 in fig. 3, an unperturbed edge. With the Hough technique, the count in the parameter space corresponding to line #5 was lower than that of line #4, and line #5 was not detected, although it is releatively a long line. However, LF transformation had a corresponding count sufficiently large for this line to be detected. In other words, LF transformation has the significant advantage of enhancing relatively long perturbed lines in digitized pictures.

Let us now consider an image of n by m pixels, and define k different quantization values for θ. In the Hough algorithm, the accumulators would be incremented kmn times, while LF algorithm would increment the accumulators mn(mn--1)/2 times. The time required to process the sample picture in fig. 3 by LF algorithm was twice that of the Hough algorithm, for k = 20 and m = n = 20. But notice that in LF transformation, the processing time necessary is independent of k. This is a second major advantage of LF algorithm, since it allows the user to have higher resolution for r or θ without substantial increase in processing time.

III. WEIGHTED LINE-FINDING ALGORITHM

The algorithm introduced in section 2 needs a large amount of processing time for large m or n, since the computation required is of order $(m \cdot n)^2$. To compensate this weakness, we present its extension, the weighted line-finding (WLF) algorithm. Take a point (X_i, Y_j) in the image and put a $(2p + 1)$ by $(2p + 1)$ window centered there. By giving different weights to the lines connecting the point (X_i, Y_j) with other points inside the window and moving the window through the image, we can find a new version of the corresponding parameter space. Through this algorithm, we can define a special criterion for lines under search by defining different size windows and different weighting factors for neighboring points. The dimensions of the window are usually small, this limits the processing time of the algorithm.

In this algorithm, the accumulators are incremented an order of mn times, which is independent of k, the number of quantization levels for θ. To implement this algorithm, for each point (X_i, Y_j) in the image space, we will consider all of the neighboring points within a $2p + 1$ by $2p + 1$ window centered at (X_i, Y_j). By defining weights W_{kl} for each point $(X_i + k, Y_j + \ell)$ inside the window, we can find the parameters of the line passing through these. Then by incrementing the corresponding accumulator in the parameter space by W_{kl}, we can process all image pixels. The program inspects the array to locate cells with high counts.

To compare this algorithm with the Hough algorithm, WLF algorithm was implemented by the first author on the PDP/10 and applied to data shown in fig. 3. By defining a 7 x 7 window and assigning a weighting factor of 4 for the first nearest neighbors of its center, a factor of 2 for its second nearest neighbors, and a factor of 1 for its third nearest neighbors, the same results have been obtained as the Hough algorithm. The time required to process the sample picture of fig. 3, by the two algorithms, was the same, and remains the same when the dimensions of the image are increased. Table 3 shows the contents of accumulator array for WLF algorithm.

Since each accumulator of parameter space will be incremented if and only if there is a corresponding line in the image, the number of zero–value accumulators in WLF algorithm is higher than that of the Hough algorithm. This means that we need less storage to store the output data if we use a data structure that permits us to store only non–zero output data. The second advantage of this algorithm is the capability of the algorithm to define and use special criteria for search for lines through introduction of the weighting factors. Moreover, the processing time is independent of k, and is an order of mn. It thus allows the user to have higher resolution for r or θ, without substantial increase in incrementing time. For an m by n image and $2p + 1$ by $2p + 1$ window, the number of accumulator increments is almost equal to $4p(p + 1)mn$.

1. Example 2

The following example illustrates some advantages of WLF over the Hough algorithm using fig. 4 as the input data. In this image line #5 is a disturbed line longer than line #4, and line #1 is a disconnected line longer than line #2. If we apply the three algorithms, i. e. LF, WLF and the Hough algorithm, and choose different threshold values sufficient to detect line #2 in each case, we see that the Hough algorithm will detect line #1, #2, #3, #4, and line #5 (line #3 and line #4 will have the same parameters due to quantization effect in parameter space). Using LF, all the lines from #1 to #5 were detected. The disconnected lines are distinguished from the connected ones by WLF where only lines #1, #3, and #4 were detected. The output results of the three algorithms are shown in Tables 4, 5, and 6. The count for line #1 there is less than that of line #2, because of the discontinuity present in line #1, this occurs although line #1 is longer than line #2.

With $m = n = 30$ for fig. 4, and using $k = 20$, the processing time for WLF was almost the same as that of the Hough algorithm, but the time required for LF was five times higher. As shown in Table 4, 5, and 6, the number of non-zero entries in Table 4 is higher than that of Table 5 and 6, so to store them, the former needs more memory. Table 6 has fewer non-zero entries than the others. This shows an important advantage of WLF algorithm.

2. Trade-off among LF, WLF and the Hough algorithms

The three algorithms, LF, WLF and the Hough transform, were compared and the results have been tabulated in Table 7. For an image of m by n pixels, the computation time is proportional to mn and kmn for WLF and the Hough algorithm respectively, while for LF, the time is proportional to $mn(mn-1)/2$. This is the main weakness of LF transforms.

In the Hough algorithm, processing time is proportional to the number of quatization levels for θ, while this is not the case for WLF and LF. This makes it possible to have a higher resolution in parameter space without higher processing time.

The storage requirement for WLF is less than that of LF and the Hough transform because there are more zero-value accumulators for WLF results.

With the WLF algorithm, it is possible to define an arbitrary criteria for search for a line. By this capability, it is possible to distinguish dashed lines and perturbed lines from connected ones. Although with the LF transform it is not possible to define a special criteria for search for lines, perturbed lines are enhanced by it. With respect to connected ones, this is an advantage of LF with respect to the Hough transform.

In the Hough transform and LF, the accumulator counts will show the number of pixels in each line, while in WLF, it is possible to associate the length of the line to the count of accumulators. This makes it possible to eliminate the sensitivity of the transform to the directional orientation of the lines.

IV. CONCLUSION

In section 2, we discussed a new technique to detect edges in an image, called LF, that is less sensitive to the line perturbations that occur in the digitization process. Furthermore, the processing time requirement of the method does not depend on the way the parameter space is quantized. In section 2.2, this algorithm was applied on a sample image and its advantages were discussed.

In section 3, an extension to the above algorithm, called WLF algorithm was presented. The main advantage of this technique is the ability to define arbitrary criteria for search for lines. This was shown in section 3.1 as well as several other advantages. While the WLF technique needs the same processing time as the Hough algorithm, it needs less memory to store the results. The processing time is independent of the number of quantization levels in the parameter space, thus for higher k (the number of quantization levels for θ), it needs less time than the Hough algorithm.

Further experimentation on real data from general images, i.e. ones with nonbinary pixels, is needed to establish the processing time, storage, and accuracy of the weighted line-finding algorithm as a road — and general mapping/charting/geodesy feature — locator.

REFERENCES

1. Duda, R. O., and Hart, P. E., "Use of Hough Transform to Detect Lines and Curves in Pictures," *CACM*, Vol. 15, Jan. 1972, pp. 11–15.

2. Hough, R. V. C., "Method and Means for Recognizing Complex Patterns," U. S. Patent #306954, Dec. 18, 1962.

3. Kimme, C., Ballard, D., and Sklansky, J., "Finding Circles by an Array of Accumulators," *CACM*, Vol. 18, Feb. 1975, pp. 120–122.

4. Shapiro, S. H., "Transformation for the Computer Detection of Curves in Noisy Pictures," *Computer Graphics and Image Processing*, 4, 1975, pp. 328–338.

APPENDIX A

Taking a line passing through a pair of points (X_1, Y_1) and (X_2, Y_2), its equation will be

$$\text{(a1)} \quad \frac{X - X_1}{Y - Y_1} = \frac{X_1 - X_2}{Y_1 - Y_2}$$

or

$$\text{(a2)} \quad Y = Y_1 + (Y_1 - Y_2) \cdot (X - X_1) / (X_1 - X_2)$$

And the slope of the line will be:

$$\text{(a3)} \quad m = (Y_1 - Y_2) / (X_1 - X_2)$$

Since at point A, $Y = O$, therefore the distance OA will be

$$\text{(a4)} \quad OA = (X_2 Y_1 - X_1 Y_2) / (Y_1 - Y_2)$$

Taking the normal to the line from the origin, the slope of the normal will be

$$\text{(a5)} \quad m = -1/m = -(X_1 - X_2) / (Y_1 - Y_2)$$

So the equation of the normal will be

$$\text{(a6)} \quad Y = mx = -(X_1 - X_2) \cdot X/(Y_1 - Y_2)$$

By equating the (a2) and (a6), we can find the coordinates of the intersection (X_i, Y_i):

$$\text{(a7)} \quad X_i = [(X_1 Y_2 - X_2 Y_1) \cdot (Y_2 - Y_1)] / [(X_1 - X_2)^2 + (Y_1 - Y_2)^2]$$

$$\text{(a8)} \quad Y_i = [(X_1 Y_2 - X_2 Y_1) \cdot (X_1 - X_2)] / [(X_1 - X_2)^2 + (Y_1 - Y_2)^2]$$

and the length of the normal to line 1, r_1, would be found from:

$$\text{(a9)} \quad r_1^2 = X_i^2 + Y_i^2$$

$$\text{(a10)} \quad r_1 = | X_1 Y_2 - X_2 Y_1 | \sqrt{(X_1 - X_2)^2 + (Y_1 - Y_2)^2} / [(X_1 - X_2)^2 + (Y_1 - Y_2)^2]$$
$$= | X_1 Y_2 - X_2 Y_1 | \sqrt{(X_1 - X_2)^2 + (Y_1 - Y_2)^2}$$

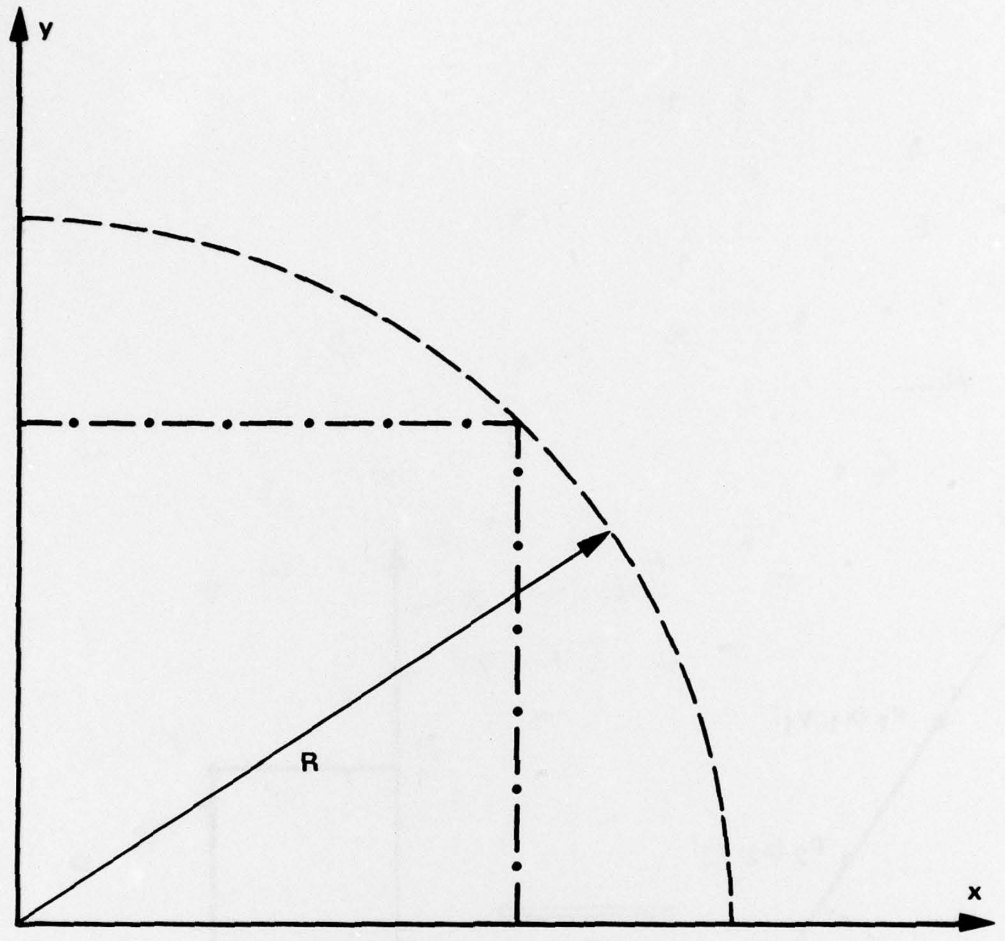

Fig. 1. Retina for a n x n Image

Key Table

Symbol	
— · — · —	Boundary for a n by n image
— — — —	Boundary for the corresponding Retina

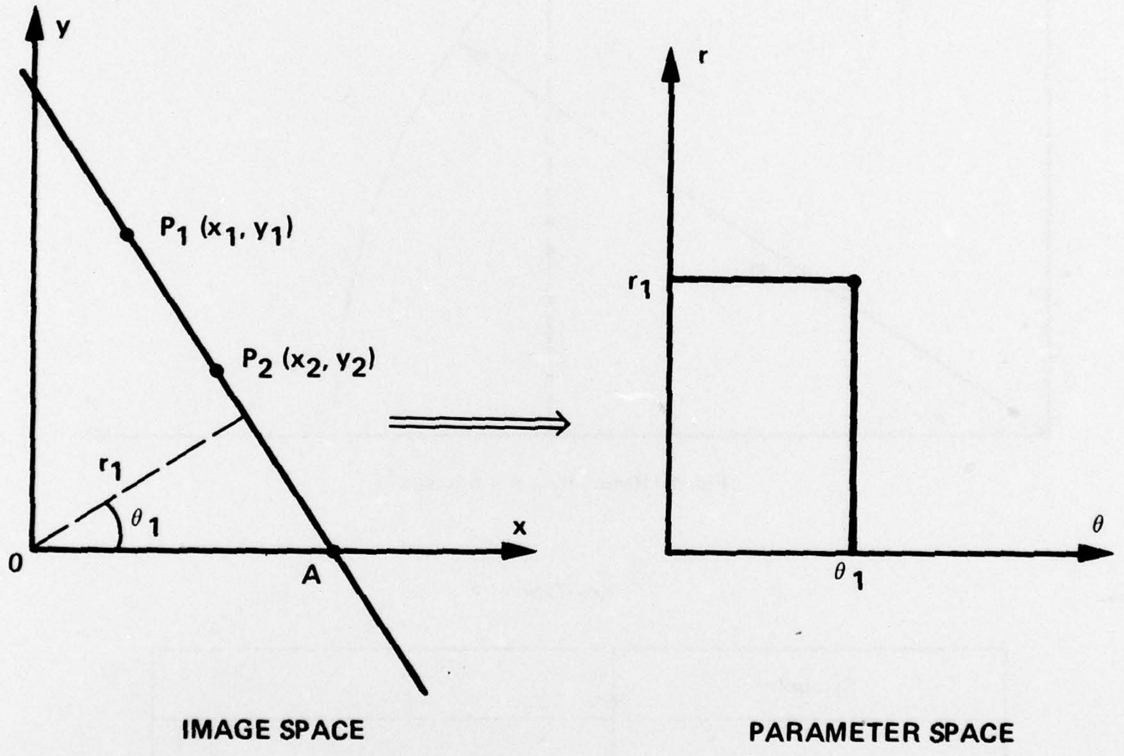

Fig. 2. The Line Transformations

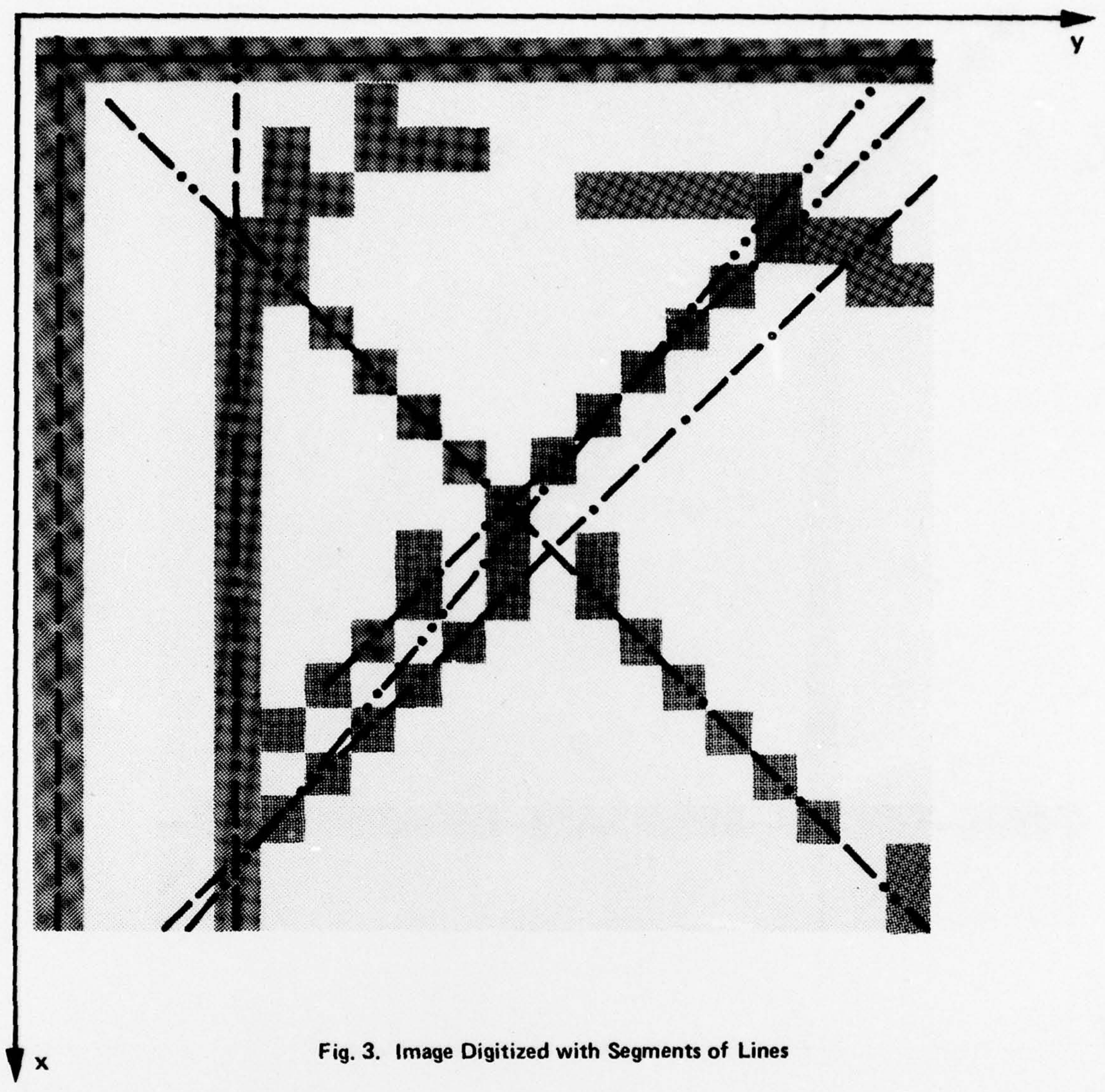

Fig. 3. Image Digitized with Segments of Lines

Key Table

Symbol	Line No.	r	θ
———————	1	1	0°
— · · — · · — · ·	2	16	45°
— · · · — · · · —	3	16	54°
— — — — —	4	1	90°
— — — · — —	5	5	90°
— — · · — — · ·	6	0	135°

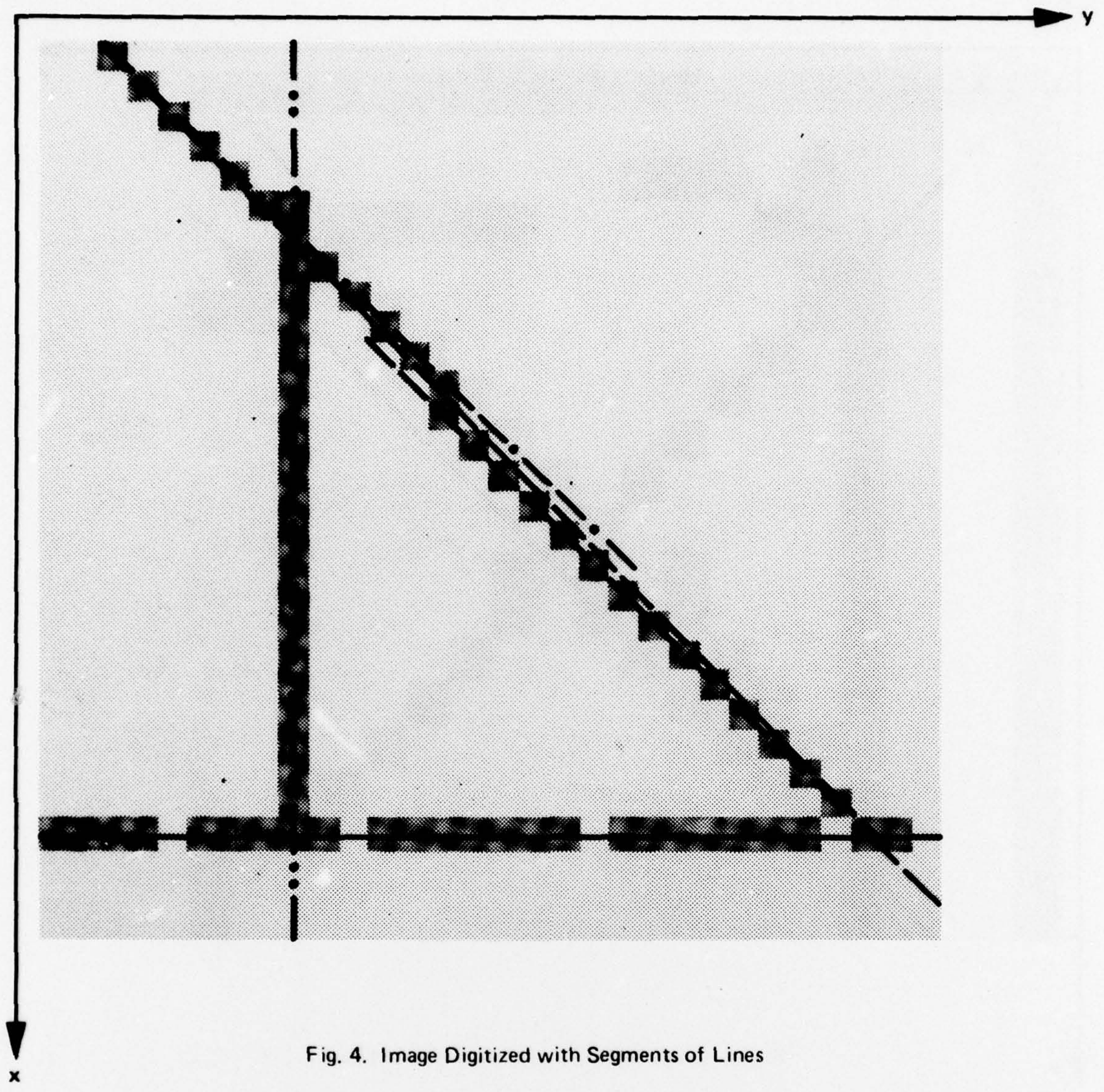

Fig. 4. Image Digitized with Segments of Lines

Key Table

Symbol	Line No.	r	θ
————————	1	27	0°
— · · — · · — · ·	2	9	90°
— — · — — · ·	3	1	135°
· — — — — —	4	1	135°
— — — — — -	5	2	135°

R	An=0	9	18	27	36	45	54	63	72	81	90	99	108	117	126	135	144	153	162	171
-28.																				
......																				
-3													3	5	5	4	7	6	5	6
-2												5	4	5	6	3	8	4	4	10
-1												6	6	7	3	6	5	3	3	7
0												6	8	6	6	15	4	5	10	9
1	20	3	1	1	1	1	1	1	1	3	20	3	6	5	5	7	3	5	12	6
2	5	7	5	3	2	2	2	3	5	7	1	6	5	4	8	3	4	6	5	5
3	5	9	4	3	3	2	3	3	4	8	1	9	7	7	9	2	6	8	3	
4	8	11	5	3	3	4	3	3	4	5	1	7	6	8	6	6	4	8	3	
5	6	4	8	6	3	2	3	4	4	1	17	7	7	2	5	2	5	3	3	
6	6	7	10	6	5	5	3	3	5	7	7	4	4	7	3	6	6	2		
7	4	4	6	10	8	5	6	7	10	11	5	7	5	6	3	4	6	2		
8	4	7	6	6	9	12	10	11	8	10	6	5	4	5	5	5	1	2		
9.																				
......																				
14	5	5	4	6	7	5	9	7	6	3	4	4	2	3	1					
15	5	6	5	4	9	5	8	7	4	5	4	4	3	3	1					
16	5	5	8	6	8	15	13	6	8	4	4	3	3	2	1					
17	4	5	4	10	7	8	3	7	3	4	4	4	3	1						
18	4	4	5	4	6	5	4	2	3	3	3	3	1							
19	3	4	4	3	1		1	2	3	3	3	2	1							
20.																				
......																				
28.																				

TABLE 1. Accumulator Array of Fig. 3 for the Hough algorithm

R	An=0	9	18	27	36	45	54	63	72	81	90	99	108	117	126	135	144	153	162	171
-28. . . .																				
. . . .																				
-2	4												6	12	12	15	33	5	6	34
-1													14	16	11	33	9	6	17	34
0												1	30	19	15	114	7	10	54	14
1	190										190	5	15	12	14	30	7	8	39	
2	17	8	2	1		1		1	2	8	7	9	14	11	29	19	5	17	18	
3	10	29	4	1	1	1	1	1	4	11		18	16	16	39	5	5	22	1	
4	28	33	11	3	1	2	1	3	5	7	7	26	20	19	15	9	6	20		
5	19	20	31	7	4	1	4	3	7	3	139	16	15	10	16	2	6	4		
6	21	15	43	16	6	6	3	4	15	25	46	12	12	23	10	6	9			
7. . . .																				
. . . .																				
14	10	7	7	25	29	37	37	34	15	7	7	7	6	2						
15	11	8	11	19	48	45	50	26	11	6	8	9	4	1						
16	11	8	17	10	40	98	98	12	14	6	9	5	2							
17	7	8	10	27	23	47	6	14	6	7	10	5	1							
18	7	7	8	9	6	13	3	3	3	4	5	3								
19. . . .																				
. . . .																				
28. . . .																				

TABLE 2. Accumulator Array of Fig. 3 for the line--finding algorithm

```
R    An=0 ' 9    18   27   36   45   54   63   72   81   90   99  108  117  126  135  144  153  162  171
-28.......................................................................................................
..........................................................................................................
-2                                                                     2    1         1
-1
0                                                                                  76             1
1      129                                          129         1              1    1             6
2       13        1    2         4         2    1                              2    7    2    4    2
3       13                  1    2    1                                   4    1         1    8
4       28        1              1                                  2    4    2   12    3    8
5       14        3    4                                101         1              2    1
6        9        4    6    1    2                       22         1    6        11    2
7.........................................................................................................
..........................................................................................................
14       3                  1    1    3         2             1         1
15       4             2    1    1    2    4    3             2         1
16       7        4    2    2   66    5         2             3
17       2        1    1   38         2    1         5
18........................................................................................................
..........................................................................................................
28........................................................................................................
```

TABLE 3. Accumulator Array of Fig. 3 for the weighted line–finding algorithm

R	An=0	9	18	27	36	45	54	63	72	81	90	99	108	117	126	135	144	153	162	171
-42.																				
.																				
-2												1	1	4	3	2	2	2	1	1
-1												1	1	3	2	4	6	3	2	2
0												1	1	3	3	1	5	3	1	
1	1	1									1		5	3	3	29	3			
2	1		1	1							1	1	3	4	5	1				
3	1	1	1	1		1	1	1	1	1	1	2	2	6	6	7				
4	1	1	1	1	1	1	1	1	1	1	2	3	6	6	11					
5	1	1		1	1					2	1	7	6	3	6					
6	2	1	1	1		1	1	1	1	1	2	8	5	4	4					
7	1	2	1		1	1	1	1	1	2	2	8	3	6	2					
8	2	1	2	1	1	1			1	2	2	4	3	3	1					
9	22		2	1	2			1	1	2	1	22	3	3	3					
10	2	1	1	1	2	1	1	1	2	6	2	2	2	3						
11.																				
.																				
25	2	2	1	6	2	4	1	2	3	2	2									
26	2	2	4	3	4	2	3	2	1	2	2									
25	7	25	5	4	4	3		2	2	2	2	1								
28		6	6	2	1	3	1	2	1	2	1									
29		8	3	2	3	2	2	1	2	1	1									
30.																				
.																				
42.																				

TABLE 4. Accumulator Array of Fig. 4 for the Hough algorithm

```
R   An=0''9  18  27  36  45  54  63  72  81  90  99  108 117 126 135 144 153 162 171
-42.........................................................................
............................................................................
-4                                                       1   5   5   11  2

-3                                                       1   1   5   13  2
-2                                                       1   6   10   8  8
-1                                                       3   4   17  4   1
 0                                                       5   4   16  1
 1                                                   1   4   6   210 3
 2                                                   6   16  43  153
 3                                           4   9   24  25  47  13
 4                                       2   2   14  20  14  27
 5                                       3   2   16  21  15  11
 6                                       2   2   22  15  8   2
 7                                       2   3   21  8   9   2
 8                                   1   3   17  7   4   9
 9                                   2   3   238 5   4   3
10                                   3   13  11  2   5   2
11                               1   10  21  5   5   1   2
12..........................................................................
............................................................................
25   2   2   1   11  1   6   1   2   1   1   1
26   3   2   9   6   6   2   2   1           1
27   308 21  11  5   5       2   1       1
28   4   14  8   4   1   2       1   1
29       6   4   2   1   3   1       1
30..........................................................................
............................................................................
42..........................................................................
```

TABLE 5. Accumulator Array of Fig. 5 for the line–finding algorithm

```
R    An=0   '9   18   27   36   45   54   63   72   81   90   99  108  117  126  135  144  153  162  171
-42.....................................................................................................
........................................................................................................
0                                                                                       1
1                                                              1              167
2
3                                                                        1
4                                                                   2    2
5                                                                   2    2
6       4                                                      1
7                                                                   4
8       4
9       2                                             143
10      1
11
12                                          1
13                1    2         4         2
14......................................................................................................
........................................................................................................
25                               4                    1
26                          1                         2
27    137              2
28                1
29......................................................................................................
........................................................................................................
42......................................................................................................
```

TABLE 6. Accumulator Array of Fig. 4 for the weighted line–finding algorithm

	Hough	LF	WLF
Processing time	kmn	mn(mn-1)	mn
Storage Requirement	higher than WLF	higher than WLF	lower storage
Choice of criteria-defining	NO	NO	YES
Content of accumulators	Number of pixels in the line	Number of pixels in the line	Number of pixels or length of the line
Enhancing perturbed lines	NO	YES	YES

TABLE 7. Comparing Hough, LF, and WLF transforms