ETL-0478

Computer strategy for detecting line features on simulated binary arrays in support of radar feature extraction

Frederick W. Rohde

AD-A203 257

November 1988

U.S. Army Corps of Engineers
Engineer Topographic Laboratories
Fort Belvoir, Virginia 22060-5546

E T L

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION: UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT: Approved for public release; distribution is unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S): ETL-0478
6a. NAME OF PERFORMING ORGANIZATION: U.S. Army Engineer Topographic Laboratories Research Institute; 6b. OFFICE SYMBOL (If applicable): CEETL-RI-A	7a. NAME OF MONITORING ORGANIZATION: U.S. Army Engineer Topographic Laboratories
6c. ADDRESS (City, State, and ZIP Code): Fort Belvoir, VA 22060-5546	7b. ADDRESS (City, State, and ZIP Code): Fort Belvoir, VA 22060-5546
8a. NAME OF FUNDING/SPONSORING ORGANIZATION: U.S. Army Engineer Topographic Laboratories; 8b. OFFICE SYMBOL (If applicable): CEETL	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code): Fort Belvoir, VA 22060-5546	10. SOURCE OF FUNDING NUMBERS: PROGRAM ELEMENT NO. 4A161102B52C; PROJECT NO. B; TASK NO. 015; WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)
Computer Strategy for Detecting Line Features on Simulated Binary Arrays in Support of Radar Feature Extraction (U)

12. PERSONAL AUTHOR(S)
Frederick W. Rohde

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Technical	FROM May 88 TO Nov 88	November 198	29

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Radar Image Analysis, Terrain Feature Extraction, Terrain Analysis

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

Line search techniques for linear features in digital radar images are developed and described. It is shown that the search techniques can be represented by codes. The codes determine the major directions of search and the removal of side branches. The testbed that is necessary to investigate and test the search techniques is described.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT: ☒ UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION: UNCLASSIFIED	
22a. NAME OF RESPONSIBLE INDIVIDUAL: E. James Books	22b. TELEPHONE (Include Area Code): (202) 355-2774	22c. OFFICE SYMBOL: CEETL-IM-T

DD Form 1473, JUN 86 — Previous editions are obsolete. — SECURITY CLASSIFICATION OF THIS PAGE

PREFACE

This work in this report was performed under DA Project, 4A161102B52C, Task B, Work Unit 015, "Automated Radar Feature Extraction Research."

The work was performed under the supervision of Mr. Lawrence A. Gambino, Director, Research Institute, U.S. Army Engineer Topographic Laboratories (ETL).

COL David F. Maune, EN, was Commander and Director, and Mr. Walter E. Boge was Technical Director of ETL during the report preparation.

CONTENTS

ILLUSTRATIONS

ILLUSTRATIONS

COMPUTER STRATEGY FOR DETECTING LINE FEATURES ON SIMULATED BINARY IMAGE ARRAYS IN SUPPORT OF RADAR FEATURE EXTRACTION

INTRODUCTION

The purpose of this study is to develop methods for line feature detection and analysis on digital radar imagery. Line features on radar images usually appear as a line of constant gray tone or as boundaries between area features. They are characterized by length, width, shape, orientation, and gray tone. Line features on radar imagery may represent, for example, highways, railroads, bridges, canals, rivers, shorelines, mountain ridges, drainage patterns, boundaries between agricultural fields, and other boundaries. For automated line feature extraction it is useful to convert the radar image first into a binary image. This can be done with image processing techniques such as edge detection, edge enhancement, thresholding and segmentation. The binary image has then to be searched for line features. One method to search for line features on imagery is to search for line paths in windows of the image. An image can be divided into windows of equal size; for example an image consisting of 512 by 512 pixels can be divided into 1024 windows of the size 16 by 16 pixels. The line path of a window starts at one side of the window and ends at another side of the window. A path of a window is a string of pixels which are connected by 4-connectivity except for the end points of the path. The path through the window can be approximated by a line element characterized by its slope and location coordinates in the window. The slopes of all line elements of the image can be organized into a slope histogram or registered as another image array. The window line elements can be analyzed for their properties of forming lines of definite length, shape, and for line connections and parallelism. This investigation deals primarily with search strategies for finding paths in image windows. The binary window arrays for the study were computer generated rather than derived from real images because they can be easily produced and modified in a controlled manner.

LINE FEATURES IN RADAR IMAGES

Figures 1 through 6 show radar images that represent different landforms and cultural features, including line features. The radar images of Figures 1 through 3 are from locations in Alabama; the images in Figures 4 through 6 are from locations in Alaska.

Figure 1 shows a location that contains mountains and a valley with cultural features. At the left side of the image one can recognize a dark line that runs from the top to the bottom of the image. This line feature is a road. In the mountain area one can detect at least two major line features running from top to bottom and representing a valley between mountain ranges. In the valley portion of the image one can recognize many line features. Most of the line features that are short and form right angles are boundaries between agricultural fields. The pronounced bright line features in the valley are lines of trees or shrubs along roads or creeks or they are boundaries between agricultural fields. The tree lines that are along roads or fields are usually straight. Those along creeks are usually not straight. The line features in Figure 2 are produced primarily by valleys, ridges, and roads. The line features of this image are not randomly distributed, but are structured according to the patterns of the landform.

In Figure 3 the most obvious line features are the two bridges across the river and shorelines of the river. The shorelines appear on a macroscale as straight lines, but show many irregularities on a microscale. The city to the right of the river shows many line features, which are caused by lines of buildings and roads. These line features have preferred directions in accordance with the city's street pattern.

Figure 4 shows line features representing shorelines, two runways of an airfield appearing as black lines in the lower right corner of the image, and straight forest boundaries. The line features in Figure 5 are produced by rivers, mountain ridges, and shadows. In Figure 6 the shoreline represents a relatively straight line feature, and the beach ridges are shown as a family of gently curved line features.

The above discussed examples of line features in radar imagery demonstrate that the lengths, orientations, distribution, and geometric relationships of line features may be used in the process of detecting, recognizing, and classifying natural and cultural terrain features.

The fact that the orientation of line features on radar imagery are not randomly distributed but show normally only a few preferred orientations led to the idea to develop line feature search strategies for preferred orientations.

INVESTIGATION

SIMULATION TESTBED. A digitized radar image can be partitioned into rectangular windows that can be represented by arrays of pixels. In this investigation the windows were selected as square arrays, where the number of rows and columns is equal. The digitized, processed image is assumed to be binary, so that the array elements are either zero or one. In order to process and manipulate the array elements a computer simulation testbed was developed.

The testbed algorithm can generate square arrays up to the size of 32 by 32 elements. The arrays can be displayed or printed. The testbed provides the capability to partition the array into fields. The elements of a field can be selected so that they are all 0's, or 1's, or 0's and 1's in a random distribution. A field may also consist of a line or a path. The random distribution of 0's and 1's is generated by a built-in random number generator. The value of individual elements can be changed by entering zeros or ones from the keyboard.

With this capability one can generate paths consisting of ones that continue from one side of the window to another side. All other elements of the window that do not belong to an established path are randomly distributed zeros and ones. Figure 7 shows an array of 32 by 32 elements with one path of ones going from the left side to the right side of the window. The path cannot be readily recognized because it is obscured by randomly distributed zeros and ones. In figure 8 all elements except the elements of the path are set to zeros. The path represented by the ones is easily recognized.

SEARCH STRATEGY. This investigation deals with the search for paths that begin at the left side of the array and end at the right side or the bottom of the array. The concepts of two search codes and two backtrack codes are developed and discussed. The search codes are called RIGHT and DOWN and are represented by the code symbols R and D. The backtrack codes are called LEFT and UP and are represented by the code symbols L and U. The codes R, D, L, U can be combined and operated in the combinations R, RD, RDL, RDLU. Code priorities and conditions to make decisions for search, backtrack, and switch are developed and discussed.

The R code searches for 1's in the first column at the left side of the array by moving from element (0,0) to the elements (1,0), (2,0) , ... until an element (j,0) = 1 is found. At this point, the R code triggered by the condition

$$\{(j,0) = 1\}$$

switches to search for 1's along row j such that (j,0) = 1, (j,1) = 1, ... until (j,k) reaches the last column on the right side of the array and the path search is completed. If R finds an element (j,k) = 0, R returns to the element (j+1,0) and starts the next search. The search strategy of the R code is discussed using an example in the following paragraph.

Figure 9 shows a square array of 12 by 12 elements. The rows of this array are counted j = 0, 1, 2,...,11 and the columns k = 0, 1, 2, ..., 11. The array contains a path that begins on the left side at the element (3,0) and ends on the right side at the element (3,11). The R code starts at element (0,0) = 0 and moves to element (3,0) = 1. The search for the row of 1's starts at (3,0) and continues along row j = 3 until the element (3,11) = 1 is reached. At this point the path search is completed. The R search moves to (4,0) and searches for 1's along the k = 0 column. At (9,0) = 1, R switches to search for 1's along the j = 9 row until the element (9,3) = 0 is found. At this point the R search of the array is completed.

The RD code starts searching using the R code. If the element (j,k) = 0 is found and the condition

$$\{(j,k) = 0 \text{ and } (j,k-1) = 1 \text{ and } (j+1,k-1) = 1\}$$

is met, the R code switches to the D code. The D code searches for 1's along the k-1 column such that the conditions

$$\{(j,k-1) = 1 \text{ and } (j,k) = 0\},\ \{(j+1,k-1) = 1 \text{ and } (j+1,k) = 0\},$$
$$\{(j+2,k-1) = 1 \text{ and } (j+2,k) = 0\},\ \ldots$$

are met until an element (j+d,k) = 1 is found. At this point, triggered by the condition {(j+d,k) = 1}, the D code switches back to the R code. R and D codes switch from one to the other if the required conditions are met. The R code has priority over the D code. Both search codes are combined into the symbol RD. The strategy of the RD code is discussed with an example in the following paragraph.

Figure 10 shows a 12x12 square array similar to that of Figure 9. The array contains a path that begins on the left side at the element (3,0) and ends on the right side at the element, (7,11). The search for the path section (3,0), (3,1), (3,2), (3,3), (3,4), (3,5) is executed by the

R code and follows the row number 3. The search for the path section (4,5), (5,5), (6,5), (7,5) is executed by the D code and follows the column number 5. The search of the path section (7,6), (7,7), (7,8), (7,9), (7,10), (7,11) is executed by the R code and follows the row number 7. At (7,11) the search for the path is completed. The search code RD begins with the R code and starts at element (0,0). At (3,0) the condition {(j,0) = 1} is met and R switches to a search for 1's along the row j = 3. At (3,5) the condition

$$\{(3,6) = 0 \text{ and } (3,5) = 1 \text{ and } (4,5) = 1\}$$

is met and R switches to D. The D code continues along the k = 5 column to element (7,5). At this point, the condition

$$\{(7,6) = 1\}$$

is met and the D code switches back to R search. The R search continues until the right side of the array at element (7,11) is reached. At this point the path search is completed. The RD search moves to element (4,0) and starts the next RD search.

The RDL code is used when horizontal dead end branches exist. The path in Figure 11 has two dead-end branches. The elements (3,6), (3,7), (3,8), (3,9) form a horizontal dead-end branch. The elements (8,5), (9,5), (10,5) form a vertical dead-end branch. If the RD code is applied to the array in Figure 11, the search will stop at element (3,10). In order to find the path, the search code RD has to be augmented by the backtrack code L. The RDL code detects at (3,10) = 0 the condition

$$\{(3,10) = 0 \text{ and } (3,9) = 1 \text{ and } (4,9) = 0\}$$

which triggers the RDL code to switch to the backtrack code L. In general the RDL switches to L if the condition

$$\{(j,k) = 0 \text{ and } (j,k-1) = 1 \text{ and } (j+1,k) = 0\}$$

is met. The backtrack code L proceeds from element (j,k) to the elements (j,k-1), (j,k-2), ... if the conditions

$$\{(j,k-1) = 1 \text{ and } (j+1,k) = 0\}, \{(j,k-2) = 1 \text{ and } (j+1,k-2) = 0\}, \ldots$$

are met. If the condition

$$\{(j+1,k-1) = 1\}$$

is found the backtrack switches to the search code D of RDL. In Figure 11 the L code backtracks from (3,9) to (3,5) where the the condition {(3,5) = 1} is found and the switch from L to D occurs. The RD code searches until the element (7,11) is reached and the path search is completed. The second dead-end branch is ignored because R search has priority over D search.

The RDLU code is used when a horizontal-vertical dead-end branch exists as shown in Figure 12. The array in Figure 12 has two dead-end branches. The R code of RDLU starts at (0,0), moves to (3,0) and continues along the row j = 3 to element (3,8) = 0. The dead-end branch formed by the elements (1,2) and (2,2) is ignored by the R code. At (3,8) the R code switches to the D code. The D code moves down along the column k = 7 to the element (6,7) = 0 where the condition

$$\{(6,7) = 0 \text{ and } (5,7) = 1 \text{ and } (5,6) = 0\}$$

is found. This condition triggers the RDLU code to switch to the U code. In general, the RDLU code switches to the U code when the condition

$$\{(j,k) = 0 \text{ and } (j\text{-}1,k) = 1 \text{ and } (j\text{-}1,k\text{-}1) = 0)\}$$

is met. The backtrack code U proceeds from element (j,k) to the elements (j-1,k), (j-2,k), ... if the conditions

$$\{(j\text{-}1,k) = 1 \text{ and } (j\text{-}1,k\text{-}1) = 0\}, \{(j\text{-}2,k) = 1 \text{ and } (j\text{-}2,k\text{-}1) = 0\}, \ldots$$

are met. If the condition

$$\{(j\text{-}u,k\text{-}1) = 1\}$$

is found, the backtrack code U switches to the backtrack L code. L and U codes switch from one to another if the required conditions are met. In Figure 12 the U code backtracks from (5,7) to (3,7) where the condition {(3,7) = 1} is found and the switch from U to L occurs. The backtrack code L proceeds from (3,7) to (3,5) where the condition {(4,5) = 1} is found and L switches to D. The RD code continues to track the path to the element (7,11) where the search is completed.

The algorithm represented by the code RDLU provides 4 codes. The R code searches only in the right direction and is trapped and terminated in right dead end branches. The RD code searches for right-down paths, ignores branches in the up and left direction, and is trapped and terminated in dead-end branches starting in the right or down direction. The RDL command searches only right-down, ignores branches in the up and left direction, eliminates dead-end branches extending in the right direction and is trapped and terminates in right-down dead-end branches. The RDLU command eliminates most dead-end branches. This search strategy is further demonstrated and discussed with a few examples.

DISCUSSION. Figure 13 shows an array of the size 12 by 12 with four paths starting at the left side of the array and ending at the element (9,11) at the right side of the array. The array is called Original Array. The four codes of RDLU are applied to this array each leading to a different path. All codes R, RD, RDL and RDLU start at the element (0,0) and search for the first element (j,0) = 1. At this point the search for the path begins.

The R starts at (2,0) = 1. The R code tests the elements (2,0) = 1, (2,1) = 1, (2,2) = 1, (2,3) = 1, (2,4) = 0. Each time a 1 is detected in the Original Array a 1 is entered at the same location of the Transformed Array. The Transformed Array has the same size as the Original

Array but all elements are initially set to zero. The R search detects (2,4) = 0 and terminates the search in the Original Array. The termination of the search triggers the RDLU algorithm to change all 1's entered into the Transformed Array back to zeros. The R search continues to search for the next (j,0) = 1 which is found at (7,0). The R search continues along row 7 and terminates at (7,9) = 0. The next (j,0) = 1 is found at (8,0). The R search continues along row 8 and terminates at (8,7) = 0. The fourth R search starts at (9,0) and ends at (9,11) where it is completed. The path from (9,0) to (9,11) is entered into the Transformed Array. The search algorithm calculates the slope of the path. Figure 14 shows the path in the Transformed Array after the R search of the Original Array is completed. The extracted path is a horizontal line with a slope of zero. The path consists of 12 elements.

The RD code starts at (2,0). The search terminates at (5,8) = 0. The next RD search starts at (7,0) = 1 and terminates at (7,10) = 0. The third RD search starts at (8,0) = 1. At (8,7) the R code switches to D code. At (9,7) the D code switches to the R code and completes the search at (9,11). Figure 15 shows the Transformed Array . If the path is replaced by a straight line from (8,0) to (9,11) the slope of this line is 0.09. The path consists of 13 elements.

The RDL code starts at (2,0) and terminates at (5,8) of the right-down-right dead-end branch. The next RDL search starts at (7,0). At (7,10) = 0 the RDL code switches to the L code. The L code backtracks and removes the right dead-end branch (7,8), (7,9). At (7,7) the L code switches to the RD code. The RD code continues to (9,11) where the path search is completed. Figure 16 shows the path in the Transformed Array. The slope of the line element approximating the path is 0.16. The path consists of 14 elements. The backtrack includes 2 elements.

The RDLU code starts at (2,0). The first right-down-right dead-end branch is backtracked by the LU code and removed. The second right dead-end branch is backtracked by the L code and removed. The RD code contineous to (9,11) where the path search is terminated. The Figure 17 shows the path in the Transformed Array. The slope of the linearized path is 0.64. The path consists of 19 elements. The backtrack includes 6 elements.

Figure 18 shows an example of a path having a dead end with multiple branches. The RDLU code is applied to this array and the path is shown in Figure 19. The slope of the linearized path is 0.64. The path consists of 19 elements. The two backtrack branches include 9 elements.

The RDLU code represents only one search strategy. Another code that also starts at the left side of the array is the DRUL code. In this code D has priority over R and U priority over L. Figure 20 shows an array to which the RD search of the RDLU code and the DR search of the DRUL code are applied. Figure 21 shows the path found with the RD search and Figure 22 the path found with the DR search.

Figure 23 shows a processed, binary radar image of the airport at Elizabeth City, North Carolina. The binary image consists of 480 x 320 pixels and is divided into 150 windows each of the size 32 x 32 pixels. The majority of windows have no paths that extend from the left side to the right side or bottom of the windows. The application of the RDLU code to the windows produced 23 line elements with a slope of 0.19 and 12 line elements with a slope of 1.15.

RULES FOR SEARCH CODES. The codes R, RD, RDL, RDLU start at the element (0,0) and the initial column (j,0) of the array. The search begins at the elements (j,0) = 1. If the search of any code is terminated or completed the code moves to the next (j,0) for the next search.

The R code searches for 1's along in a row in the right direction. If an element (j,k) = 0 is detected the search is terminated. The search is completed when the left column of the array is reached. The R code ignores side branches.

The RD code combines the codes R and D. R has priority over D. The RD code searches for 1's along a row in the right direction and along columns in the down direction. Specific conditions trigger the RD code to switch from R to D, from D to R, to terminate the search, or to complete the path search. These conditions are discussed in the section Search Strategy. The RD code ignores dead end and side branches in the left and up direction.

The DRL code combines the codes R, D, and L. The R and D codes are the search codes of RDL, the L code is the backtrack code of RDL. The L code backtracks along a row section of a dead end branch in the left direction. Specific conditions trigger the RDL code to switch from R to D, from D to R, from R to L, from L to D, to terminate the search, or to complete the path search. These conditions are discussed in the section Search Strategy. The RDL code ignores dead end branches in the up direction and side branches in the left and up direction.

The RDLU code combines the codes R, D, L and U. The R and D codes are the search codes of RDL, the L and U codes are the backtrack codes of RDLU. L has priority over U. The U code backtracks along a column section of a dead end branch in the up direction. Specific conditions trigger the RDLU code to switch from R to D, from D to R, from R to L, from L to D, from L to U, from U to L, to terminate the search, or to complete the path search. These conditions are discussed in the section Search Strategy. The RDLU code ignores side branches in the left and up direction.

The code RDLU can easily be changed to other search codes by permutations of the codes R, D, L, U and defining the initial column or initial row. The search from the top to the bottom or to the right side of the array can be accomplished by the code DRUL and the initial row (0,k). The search from the left side to the right side or to the top of the array can be accomplished by the code LURD and the initial column (j,n-1), where n x n is the size of the array. Code priorities and conditions for search, backtrack and switch are different for each code.

SUMMARY. Cultural and natural terrain features frequently exhibit line features with preferred orientations. Algorithms for directional search of line features are developed and discussed. The search strategy includes two forward search codes (R and D) and two backtrack codes (L and U). The codes can be combined to RD, RDL and RDLU providing capabilities for determining parameters such as line element slope, path lengths, path end points and number of backtrack elements. These parameters are important to feature analysis and automated feature extraction.

CONCLUSIONS

The following conclusions are reached:

Using directional search strategies a binary image can be transformed into an array of windows that contain only paths from one side to another side of the windows.

The path of a window can be approximated by a line element that is characterized by its slope and the end points on the window sides.

The algorithms for the directional search strategy calculate the slope of the line elements, starting and ending points of the path, lengths of the paths and dead end branches in terms of number of window elements.

The Transformed Array provides quantitative information such as length and orientation of lines, parallelism, angles of intersecting lines and branching.

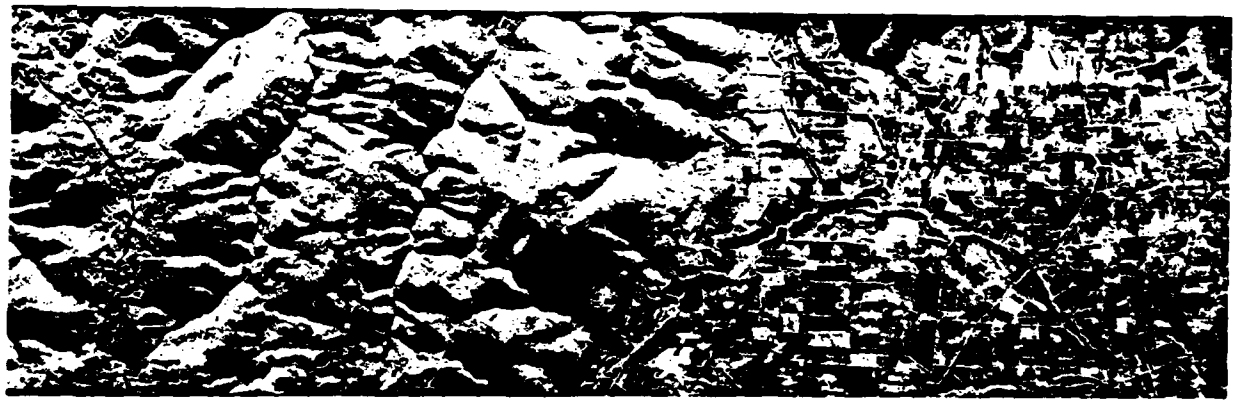


Figure 1. Mountains and Valley with Cultural Features.

Figure 2. Hilly Country with Cultural Features.

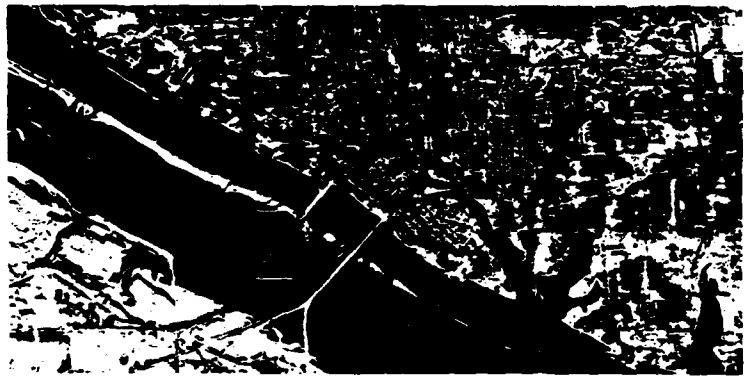


Figure 3. City and River with Bridges.

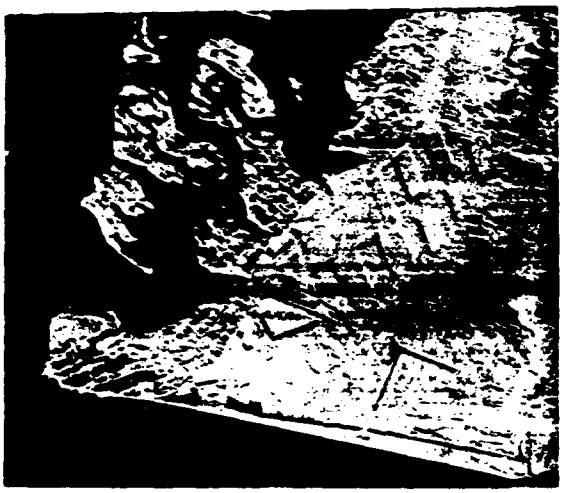


Figure 4. Coastal Area Vegetation and Manmade Terrain Features.

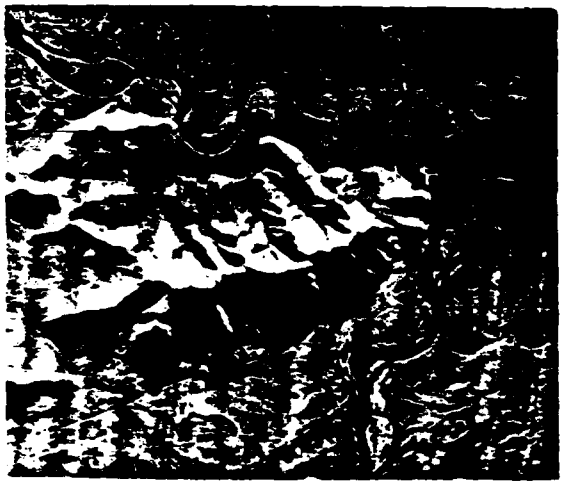


Figure 5. Plain with Rivers and Ridged Mountain.

Figure 6. Shore Line and Beach Ridges.

ORIGINAL ARRAY

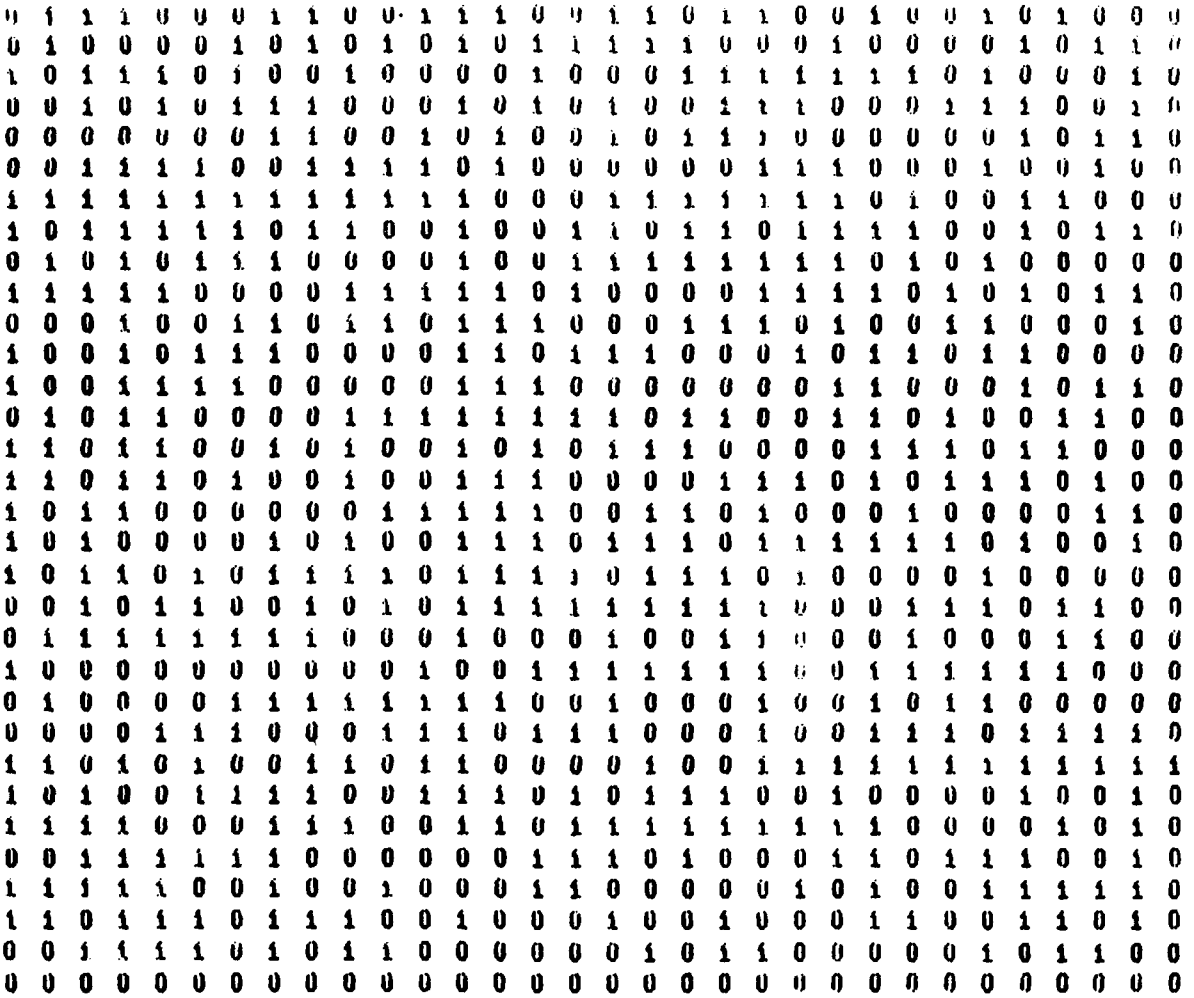


Figure 7. Window Array with One Path Going from the Left Side to the Right Side of the Array. The path is obscured by randomly distributed 0's and 1's which may be representative for other features or processing noise.

LINE STARTS: ROW 1 COL 7
LINE ENDS: ROW 32 COL 25

SLOPE OF THE LINE = .58

TRANSFORMED ARRAY

ORDERED R D L U

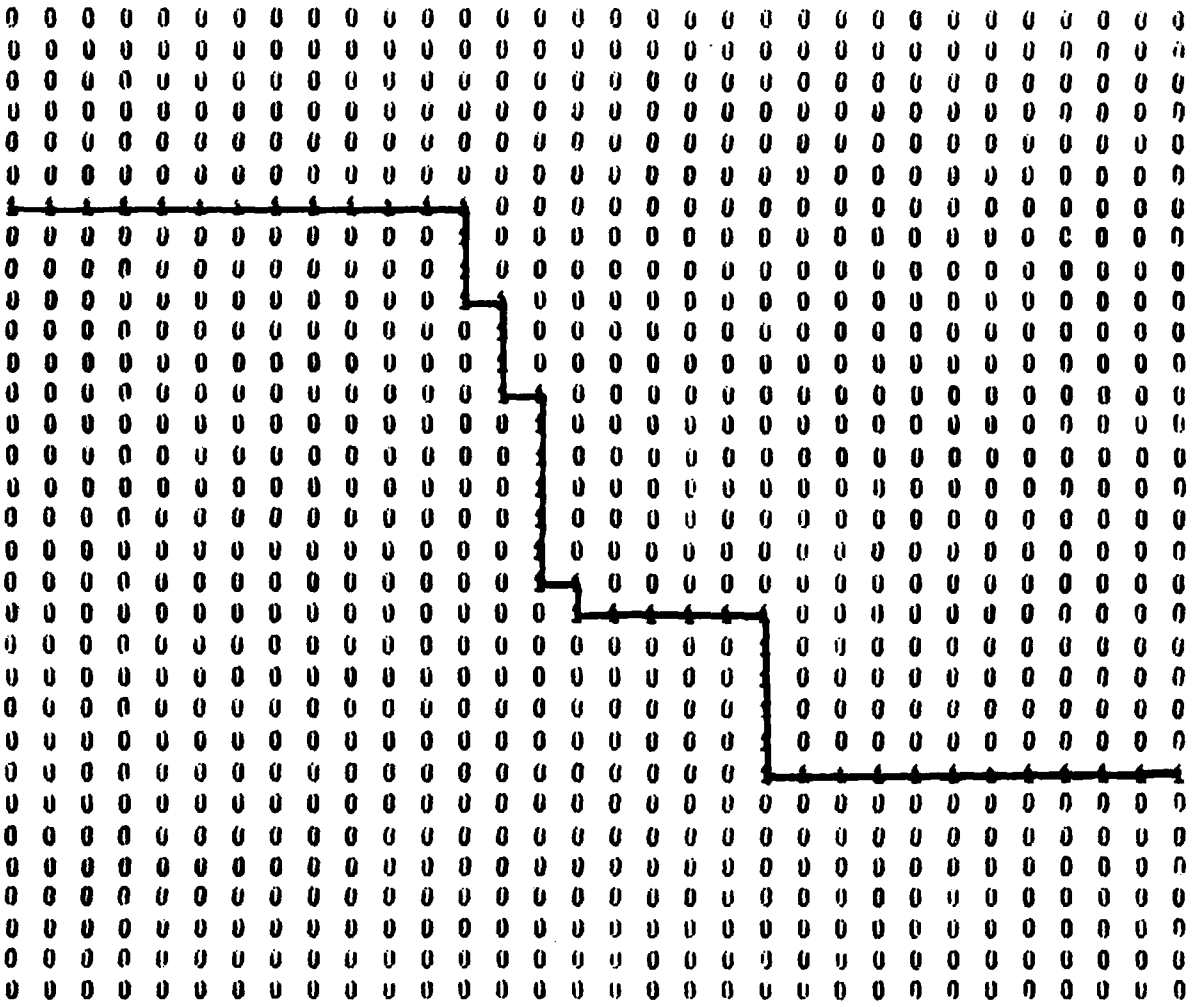


Figure 8. The Same Window Array as in Figure 7 with the Unobscured Path. All path elements are ones. All other elements are zeros.

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

Figure 9. Array with Right Path.

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

Figure 10. Array with Right-Down Path.

```
0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1 1 1 0 0
0 0 0 0 0 1 0 0 0 0 0 0
0 0 0 0 0 1 0 0 0 0 0 0
0 0 0 0 0 1 0 0 0 0 0 0
0 0 0 0 0 1 1 1 1 1 1 1
0 0 0 0 0 1 0 0 0 0 0 0
0 0 0 0 0 1 0 0 0 0 0 0
0 0 0 0 0 1 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0
```

Figure 11. Right-Down Path with Two Straight Dead-End Branches.

```
0 0 0 0 0 0 0 0 0 0 0 0
0 0 1 0 0 0 0 0 0 0 0 0
0 0 1 0 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1 0 0 0 0
0 0 0 0 0 1 0 1 0 0 0 0
0 0 0 0 0 1 0 1 0 0 0 0
0 0 0 0 0 1 0 0 0 0 0 0
0 0 0 0 0 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0
```

Figure 12. Right-Down Path with Right-Down Dead-End Branch.

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

Figure 13. Original Array -- The Array Has Four Paths Starting at the Left Column and Ending at the Right Column.

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

Figure 14. Transformed Array -- R Command Applied to Original Array.

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

Figure 15. Transformed Array -- RD Command Applied to Original Array.

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

Figure 16. Transformed Array -- RDL Command Applied to Original Array.

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

Figure 17. Transformed Array -- RDLU Command Applied to Original Array.

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

Figure 18. Original Array -- Dead End with Multiple Branches.

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

Figure 19. Transformed Array -- RDLU Command Applied to Original Array.

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

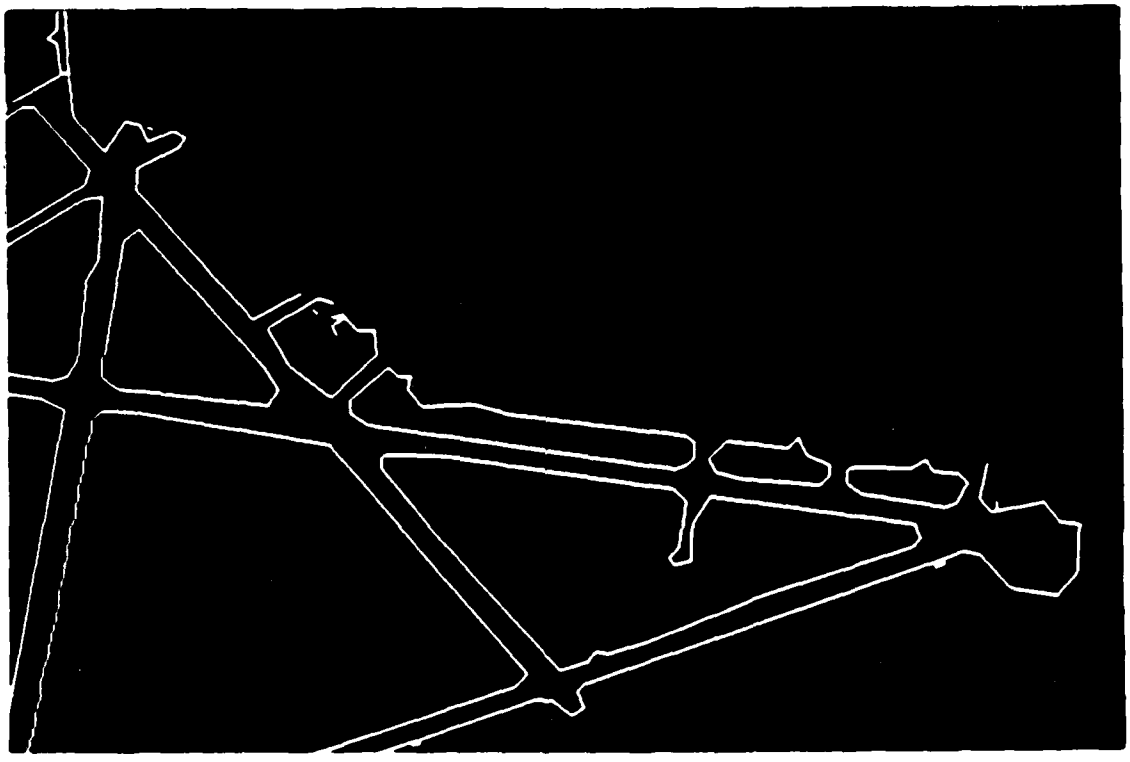
Figure 20. Original Array -- Array with Paths Starting at the Left Side and Ending at the Right and Bottom Side of the Array.

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

Figure 21. Transformed Array -- RD Command Applied to Array in Figure 20.

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

Figure 22. Transformed Array -- DR Command Applied to Array in Figure 20.



Processed by Richard A. Hevenor, USAETL

Figure 23. Binary Radar Image of Airport Run Ways.