NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

Technical Report 32-1288

A Pathfinding Algorithm for a Myopic Robot

Larry Y. Lim

JET PROPULSION LABORATORY
CALIFORNIA INSTITUTE OF TECHNOLOGY
PASADENA, CALIFORNIA

August 1, 1968

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

Technical Report 32-1288

A Pathfinding Algorithm for a Myopic Robot

Larry Y. Lim

Approved by:

[illegible]

W. F. Scott, *Manager*
Flight Computers and Sequencers Section

JET PROPULSION LABORATORY
CALIFORNIA INSTITUTE OF TECHNOLOGY
PASADENA, CALIFORNIA

August 1, 1968

TECHNICAL REPORT 32-1288

Prepared Under Contract No. NAS 7-100
National Aeronautics & Space Administration

Acknowledgment

The author wishes to thank W. F. Scott and J. J. Wedel for providing the opportunity to work on the pathfinding problem. Stimulating discussions with Dr. C. L. Lawson, Dr. G. Paine, D. Rennels, J. Rohr, E. Suggs, W. Garrison, and C. Carl are acknowledged by the author. P. Sobel provided the computer program used to generate the perspective drawing of contour configurations. In addition, the guidance provided by J. J. Wedel and D. K. Rubin is gratefully acknowledged.

Page intentionally left blank

Contents

Figures

Abstract

A pathfinding algorithm has been developed for an autonomous myopic roving vehicle. With the assumption that a path exists between two points, two propositions are proved. Proposition 1 is concerned with no obstacles and proposition 2 is concerned with at least one obstacle. Various conditions are established and terms defined. Three algorithms, necessary to direct the robot, are main, left scan, and right scan. The decisions for the navigation, control, and obstacle avoidance of the rover are based mainly on myopic (local) information of the terrain. The use of Gaussian density functions to simulate topography is considered to be novel. A FORTRAN IV computer program was written to implement the pathfinding algorithm, and the results of the computer–program runs were satisfactory.

A Pathfinding Algorithm for a Myopic Robot

I. Introduction

It has been more than ten years since E. F. Moore published his article on pathfinding algorithms (Ref. 1). In 1961, C. Y. Lee published his paper on an algorithm for path connections (Ref. 2). The algorithms discussed in the two references have three basic assumptions in common: (1) the map is completely known, (2) each path can be retraced, and (3) an optimum path can be determined. The question arises as to what degree of a priori information about the map must be known in order to determine a path from one point to another point such that it will satisfy a given set of constraints. Here, one may not be interested in obtaining the shortest path, but rather a safe path from one point to another point (i.e., if such a path exists). The impetus for the above question stems from the application of pathfinding algorithms to unmanned planetary exploration, such as the navigation and control of an autonomous roving vehicle on the surface of Mars to explore its topography. It is assumed that various orbital reconnaissances, similar to that of the *Lunar Orbiter*, have been performed and a gross knowledge of the topography of the planet is available. Consequently, the landing site and the various interesting target areas to be visited can be determined.

The transmission of the orbital reconnaissance information from a distant planet may take many weeks to accomplish because of limited transmitter power. Because of this limitation, if one is to land a roving vehicle on a distant planet, the continuous direct control of the vehicle from earth, as in the case of the *Surveyor*, will be extremely time-consuming and will probably prove to be infeasible. Consider the case of the *Mariner* Mars 1964 mission. Each television picture consisted of a 200- × 200-element matrix in which each element consisted of six bits. Since the transmission data rate was 8-1/3 bits/s, it took eight hours to assemble one television picture. Furthermore, direct control from earth of the vehicles' motion on distant planets will be extremely difficult because of transport lag in the control loop (Ref. 3). This makes the use of direct control of the vehicle from earth prohibitive. Therefore, the roving vehicle should be autonomous, with minimal command requirements for control of motion from earth.

It is the purpose of this report to discuss the results of a study involving a pathfinding algorithm for an autonomous roving vehicle. The algorithm requires only information that is within the field-of-view of the vehicle (i.e., local information). Essentially, the study involves designing a pathfinding algorithm for a myopic robot, the requirements of which are as follows:

(1) The robot shall see the terrain that is within its scanning range.

(2) Any obstacles within the robot's scanning range may be sensed.

(3) The initial starting coordinates (the landing point) shall be given.

(4) The target coordinates to be visited shall be given.

(5) If a path exists, the algorithm shall find the path.

(6) The algorithm shall avoid hazardous obstacles and be able to crawl on and over safe obstacles.

(7) The algorithm shall use a minimal number of memory locations for the navigational path.

II. Mathematical Development

The pathfinding algorithms developed in this report are based upon two propositions:

(1) If a path exists from the point P_0, (X_0, Y_0), to the point P_n, (X_n, Y_n), and there are no obstacles between the two points, then the path connecting these two points can be found.

(2) If a path exists from the Point P_0, (X_0, Y_0), to the point P_n, (X_n, Y_n), and at least one obstacle exists between the two points, then following either the right or left contour of the obstacle and always heading toward the target point P_n will result in determining a path between P_0 and P_n.

The above propositions assume that the obstacles have contour lines and that each obstacle has finite dimension (i.e., no infinitely thin obstacles). The proof of the above two propositions is based on whether a target distance sequence contains an element that can be made arbitrarily small or not. If such an element exists, then the target point is reached. Otherwise, the algorithm, which makes use of the above propositions, will result in limit-cycle oscillation.

Definition 1. An acceptable point is defined as a point (X,Y) such that $|F(X,Y)| \leq E$, where $F(X,Y)$ is a continuous function that describes the terrain and E is the elevation limit.

Definition 2. A path is defined as a route that connects two points (X_i,Y_i) and (X_j,Y_j) such that the elevation of each point of the route is $|F(X,Y)| \leq E$.

Definition 3. An obstacle is defined as a finite bounded region of a function whose amplitude or values are greater than E, where $|F(X,Y)| > E$.

A. Proposition 1

If there exists a path from the point P_0, (X_0,Y_0), to the point P_n, (X_n,Y_n), and there are no obstacles between the points P_0 and P_n, then the path can be found by the algorithm.

Proof:

Let the first intermediate point of travel be (X_1,Y_1) such that

$$X_1 = X_0 + r^*\cos\theta_1$$

$$Y_1 = Y_0 + r^*\sin\theta_1$$

where r is a fixed distance increment (scanning range) and

$$\theta_1 = \tan^{-1}\left(\frac{Y_n - Y_0}{X_n - X_0}\right)$$

In general,

$$X_{i+1} = X_i + r^*\cos\theta_{i+1}$$

$$Y_{i+1} = Y_i + r^*\sin\theta_{i+1}$$

where

$$\theta_{i+1} = \tan^{-1}\left(\frac{Y_n - Y_i}{X_n - X_i}\right)$$

Since there are no obstacles between P_0 and P_n, then every point on the line P_0P_n is an acceptable point and $\theta_i = \theta_1$ for all i.

Let the target distance sequence be $\{d_1, d_2, \cdots, d_n\}$

$$d_k = [(X_n - X_{k-1})^2 + (Y_n - Y_{k-1})^2]^{1/2}$$

By proper choice of r, the final d_k can be made as small as one quantized r (i.e., $d_k \leq r$). Hence, the target is reached.

Proposition 1 essentially states that, if the starting point (X_0,Y_0) and the target point (X_n,Y_n) have the property that $|F(X_0,Y_0)| \leq E$ and $|F(X_n,Y_n)| \leq E$, and there are no obstacles between these two points $|F(X,Y)| \leq E$ for all (x,y) between these two points, then we can determine a path that connects these points. The idea of proposition 1 is rather obvious. However, the introduction and use of the target distance sequence in proving proposition 1 is the main reason for the detailed proof. Proposition 2 will also make use of the target distance sequence argument.

B. Proposition 2

If a path exists from the point P_0, (X_0,Y_0), to the point P_n, (X_n,Y_n), and at least one obstacle exists between the two points, then following either the right or left contour of the obstacle and always heading toward the target, P_n, will result in determining a path P_0P_n.

Proof:

This proposition will be proven by the use of mathematical induction. It will be shown that the proposition is true for one obstacle between P_0 and P_n. If it is assumed to be true for m obstacles, then it can be shown to be true for $m + 1$ obstacles between P_0 and P_n. Let the one obstacle between P_0 and P_n have the contour represented by $F(X,Y)$. Let the first point of travel from (X_0,Y_0) to (X_1,Y_1) be P_1, such that

$$X_1 = X_0 + r^*\cos\theta_1$$

$$Y_1 = Y_0 + r^*\sin\theta_1$$

where r (the scanning range) is the distance increment,

and $\theta_1 = \tan^{-1}\left(\dfrac{Y_n - Y_0}{X_n - X_0}\right)$

The point (X_0,Y_0) has a neighborhood that contains some acceptable points, because the assumption of a path exists between the points P_0 and P_n. Let the acceptable contour elevation limits be given as E. Evaluate $F(X_1,Y_1)$. If $|F(X_1,Y_1)| \leq E$, then (X_1,Y_1) is an acceptable point. Now replace (X_0,Y_0) by (X_1,Y_1), then proceed to evaluate (X_i,Y_i) for $i = 2, 3, \cdots$, etc. However, if $|F(X_1,Y_1)| > E$, replace θ_1 by $\theta_1 \pm\Delta\theta$. If $\theta_1 + \Delta\theta$ is used in determining (X_1,Y_1), the left contour is being followed. If $\theta_1 - \Delta\theta$ is used, then the right contour is being followed. Evaluate the new $F(X_1,Y_1)$ by varying $\Delta\theta$, until an acceptable point is determined. Also, determine the distance from (X_1,Y_1) to (X_n,Y_n) i.e.,

$$d_1^2 = (X_n - X_1)^2 + (Y_n - Y_1)^2$$

The angular increment and decrement in $\Delta\theta$ is 1 deg.

The scanning process is continued until a first acceptable point is found. This point also determines whether the right or left contour is to be followed. The crawling process is repeated and a target distance sequence $\{d_1, d_2, \cdots, d_n\}$, is generated until an acceptable point (X_k,Y_k) such that

$$d_j > d_{k+1} = [(X_n - X_k)^2 + (Y_n - Y_k)^2]^{1/2}$$

is satisfied. Term d_j was the minimal distance from the target to (X_j,Y_j), the point where the obstacle was first encountered and $k > j$. From the point (X_k,Y_k) to (X_n,Y_n), we can apply proposition 1. If the target has not been reached after the use of proposition 1, the above scanning is repeated until the target is reached. Therefore, proposition 2 is true for one obstacle.

Assume that proposition 2 is true for m obstacles. We will have to prove that the proposition is also true for $m + 1$ obstacles. Let the intermediate point of travel from (X_i,Y_i) to (X_{i+1},Y_{i+1}) be P_{i+1}, such that

$$X_{i+1} = X_i + r^*\cos\theta_{i+1}$$

$$Y_{i+1} = Y_i + r^*\sin\theta_{i+1}$$

$$\theta_{i+1} = \tan^{-1}\left(\frac{Y_n - Y_i}{X_n - X_i}\right)$$

The expression X_i,Y_i is the point where it departed from obstacle m and encountered the $m + 1$th obstacle. Term d_i is the minimal distance from the target to (X_i,Y_i). Let the acceptable contour elevation limit be the same as before, E. Evaluate $F(X_{i+1},Y_{i+1})$. If $|F(X_{i+1},Y_{i+1})| \leq E$, then (X_{i+1},Y_{i+1}) is an acceptable point. Now replace (X_i,Y_i) by (X_{i+1},Y_{i+1}). Then, proceed to evaluate (X_{i+2},Y_{i+2}), etc. However, if $|F(X_{i+1},Y_{i+1})| > E$, replace θ_{i+1} by $\theta_{i+1} \pm \Delta\theta$. Evaluate the new $|F(X_{i+1},Y_{i+1})|$ until an acceptable point is determined. Also, determine whether $d_{i+1} < d_i$, i.e.,

$$(X_n-X_{i+1})^2 + (Y_n-Y_{i+1})^2 < (X_n-X_i)^2 + (Y_n-Y_i)^2$$

If after some P iterations, $d_p < d_i$, and there are no obstacles between P_p and P_n, we can apply proposition 1. In general, there is a monotonic decreasing subsequence of $\{d_1, d_2, \cdots, d_n\}$, say $\{d_{m1}, d_{m2}, \cdots, d_{mn}\}$, where each d_{mi} is a local minimum distance to the target such that $d_{mi+1} < d_{mi}$ for all i. Hence, the target is reached. Therefore, proposition 2 is true for $m + 1$ obstacles.

The proof of proposition 2 can be simplified by use of inductive argument on each obstacle grossly, rather than point by point. Any obstacles that merge together can be considered as a single obstacle. The value of the target distance sequence element controls the use of either proposition 1 or proposition 2. Consider the case of a box canyon. Let d_k be the distance from the end of the box canyon (Fig. 1) to the target. Assume that the right contour is being followed. Then all target distance sequence elements d_{k+1}, d_{k+2}, $\cdots$, generated will be greater than d_k. Thus, the right contour is being followed until a $d_m < d_k$ is found and proposition 1 is applied. Then, d_k is replaced by d_m. This iteration process is continued until the target is reached.

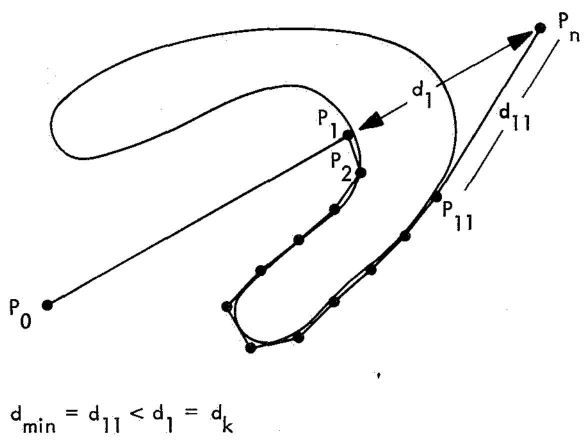


Fig. 1. Box canyon problem

III. The Pathfinding Algorithm

The pathfinding algorithm consists of three algorithms, which are called main algorithm, left scan algorithm, and right scan algorithm. The main algorithm directs the robot to move either straight ahead, in the right direction, or in the left direction. The left scan algorithm directs the robot to move in the left direction only. The right scan algorithm can only direct the robot to go to the right. The right and left scan algorithms enable the robot to navigate out of box canyons.

A. Main Algorithm

The main algorithm can be described by the following steps:

(1) Construct a vector, called target vector, by using the starting point (X_0, Y_0) and the target point (X_t, Y_t).

(2) Determine the azimuth of the target vector and call it θ.

(3) Let the next point of travel be (x,y) such that

$$x = X_0 + r^*\cos\theta$$

$$y = Y_0 + r^*\sin\theta$$

where r is the maximum scanning range of the robot and θ is defined the same as step (2).

(4) Evaluate the contour map $F(x,y)$ at the next point of travel (x,y). If $|F(x,y)|$ is within a prescribed contour limit, i.e., $|F(x,y)| \leq E$ is defined as the elevation limit (Definition 1), go to step (6); if not, go to step (5).

(5) Start scanning by varying the azimuth θ in increments of 1 deg (i.e., $\theta \pm 1$ deg) until an acceptable contour limit is reached. The scanning process scans 1 deg to the left of the azimuth, then 1 deg to the right; 2 deg to the left, then 2 deg to the right, etc. If at (x,y), $F(x,y)$ has an acceptable contour limit, go to step (6). If no such point exists, go to step (8).

(6) Determine whether the target has been reached or not. If the target has not been reached, replace (X_b, Y_b) with (X_0, Y_0), where (X_b, Y_b) is the previous point. Also replace (X_0, Y_0) with (x,y). If

$$x = X_0 + r^*\cos\theta$$

$$y = Y_0 + r^*\sin\theta$$

is an acceptable point, set the right counter, CTR = 0.0, and the left counter CTL = 0.0. If the point

$$x = X_0 + r^*\cos(\theta + \Delta\theta)$$

$$y = Y_0 + r^*\sin(\theta + \Delta\theta)$$

is an acceptable point, set CTR = 1.0 and compute

$$D_t = [(x_n - x)^2 + (y_n - y)^2]^{1/2}$$

If the point

$$x = X_0 + r^*\cos(\theta - \Delta\theta)$$

$$y = Y_0 + r^*\sin(\theta - \Delta\theta)$$

is an acceptable point, set CTL = 1.0 and compute

$$D_t = [(x_t - x)^2 + (y_t - y)^2]^{1/2}$$

Go to step (7). If the target has been reached, the job is finished. If both right and left counters are not 1.0, go to step (1).

(7) If both counters are 1.0, the robot is trapped. If the distance from the right point to the target is shorter than the distance of the left point to the target, go to the right scan algorithm. Otherwise, go to the left scan algorithm.

(8) The robot is trapped. In this case, there exists no path from point P_0 to P_t. Hence, there exist no acceptable points in the neighborhood of P_0. The robot may have landed on the peak of a mountain.

B. Right Scan Algorithm

Because the left scan algorithm is similar to the right scan algorithm, it is sufficient to describe the right scan algorithm. In order to force the algorithm to crawl along the contour edge, a crawling vector is defined. The right scan algorithm can be described by the following steps:

(1) Construct a target vector using (X_0,Y_0) and (X_t,Y_t). Also construct a crawling vector using (X_b,Y_b) and (X_0,Y_0). The minimal distance from (x,y) to (X_t,Y_t) at the time of encountering the obstacle is $D\text{min} = D_t$.

(2) Determine the azimuth of both the target vector and the crawling vector. Call the crawling vector azimuth $\theta 1$.

(3) Let the next point of travel be (x,y) such that

$$x = X_0 + r^*\cos(\theta 1 + 90°)$$

$$y = Y_0 + r^*\sin(\theta 1 + 90°)$$

where r is the maximum scanning range and $\theta 1$ is defined the same as step (2).

(4) Evaluate the contour map $F(x,y)$ at (x,y). If $|F(x,y)|$ at (x,y) is within the prescribed contour limit, go to step (6); if not, decrement $\theta 1$ by one angular degree until an acceptable contour value of $F(x,y)$ is obtained. The maximum decrement is 270 deg. Once an acceptable (x,y) has been obtained, go to step (5).

(5) Determine the distance from (X,Y) to (X_t,Y_t) and call it Dt. Replace (X_b,Y_b) with (X_0,Y_0); and, (X_0,Y_0) with (x,y). Then go to step (6).

(6) If $D\text{min} \leq D_t$, go back to step (1). However, if $D\text{min} > Dt$, go to step (7).

(7) Evaluate $F(x,y)$. If $|F(x,y)| > E$, go back to step (1). However, if $|F(x,y)| \leq E$, reset the left and right counter to zero and return to the main algorithm. The point (x,y) is

$$x = X_0 + r^*\cos\theta$$

$$y = Y_0 + r^*\sin\theta.$$

C. Simulation of Topography

In the course of developing an algorithm for the myopic robot, the problem encountered of inputting data to the algorithm (for computer simulation) was difficult because the data had to meet requirement (2) stated in Section I. Since the input data was to be topography, the data should have various levels of contours. A novel solution for simulation of terrain was made by using Gaussian density functions. Using the Gaussian density functions, one can simulate almost all terrain simply by summing a series of "Gaussian hills." The locations of these Gaussian hills can be fixed by assigning the means, μ_x and μ_y, the desired coordinates. The shapes of the Gaussian hills can be controlled by varying the value of the standard deviations, σ_x and σ_y. The height of the Gaussian hills can be varied by using an appropriate constant (positive and negative) multiplier of the Gaussian density function.

Since the use of a set of Gaussian hills to simulate topography will result in a very smooth surface, the approximated terrain will not represent an exact topography. However, this technique is so simple to use, all terrain considered in this report is made of Gaussian hills. The Gaussian hill technique can also be used to classify and sense obstacles by covering each obstacle with a Gaussian hill. Figure 2 shows a contour configuration that can be constructed by using Gaussian hills. Figure 3 is a perspective drawing of the contour configuration of Fig. 2. Figure 4 shows a maze constructed by use of Gaussian density functions. Figure 5 enables one to calculate and construct any configuration required for the simulation of topography.

D. Computer Program for the Pathfinding Algorithm and Results

A computer program was coded in FORTRAN IV language as implemented in the IBM 7090/94 IBJOB System. The flow charts for the pathfinding algorithms are shown in Figs. 6–8. The computer program, which plots the contour lines of the obstacle, was developed by Dr. Charles L. Lawson (Ref. 4) of the Computation and Analysis Section at Jet Propulsion Laboratory. The pathfinding computer program was organized so that the contour lines of the obstacle were plotted first, and then the path from the starting point to the target was plotted. Various configurations of obstacle courses were made and the robot satisfactorily navigated through each course.

Figure 9 shows a simple obstacle course of five Gaussian hills. The path, which connects the starting and target points, satisfies the contour elevation constraint of $E = 1$ (i.e., $|F(x,y)| \leq 1.0$). The scanning range r is set at 0.1 unit. This means that the robot will see no further than 0.1 unit ahead.

Figure 10 shows a simple "box canyon" configuration. In this case, there are two paths to the target point. The path determined by the algorithm is certainly not the shortest path. The shortest path was not chosen, because the decision to go to the right scan program was based on the minimum azimuth scanning angle to the right. Also, the distance from the right scanned point to the target point was shorter than the distance to the left scanned point. Using local information only, there are no methods by which one can determine the shortest path between two points.

Figure 11 shows a maze in which the starting point is outside of the maze and the target point is inside of the maze. Figure 12 shows the same maze as Fig. 11; however, the starting point is inside the maze and the target point is outside of the maze. In all cases which were run on the computer, paths were found for each case.

IV. Conclusion

This report has demonstrated that it is possible to find a path connecting two points on a continuous contour map, using only local information. The algorithm used to find a path is based on two propositions. In turn, these propositions were based on intuitive ideas that are used in unknown areas. One of the assumptions used in proving the propositions was that there exists at least one path from the starting point to the target point. This is not an unreasonable assumption. If one were to land a spacecraft inside a crater and the wall of the crater was very steep, such that the vehicle could not extricate itself from the crater, one would have to terminate the mission.

The algorithm developed in this report might be improved by incorporating certain local terrain information, such as the hidden lines of obstacles (Ref. 5). The slopes of local areas can also be incorporated so that the vehicle will have minimal tilt angles while it travels from one point to another point. Furthermore, if we define a subpath as $|f(x_i,y_i) - f(x_{i-1},y_{i-1})| \leq E$ for all i's and for a distance of r, then the algorithm will allow the robot to crawl on and over hills. Figure 13 shows the result of the robot crawling on and over hills.

The present scanning portion of the computer program is based on simple strategy. Future work will incorporate the scanning strategy of ranging radar.

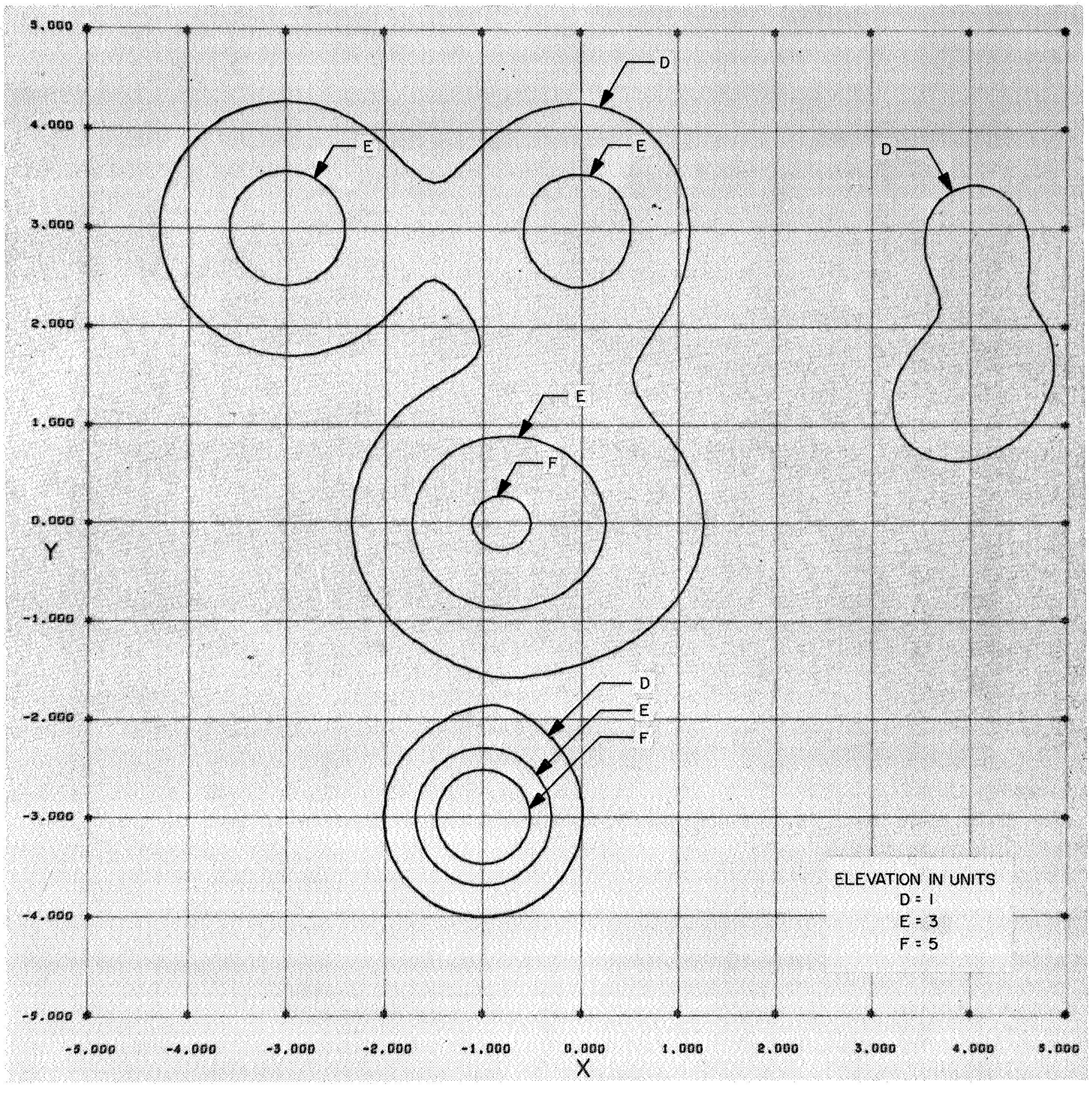


Fig. 2. Contour configuration constructed by use of Gaussian hills

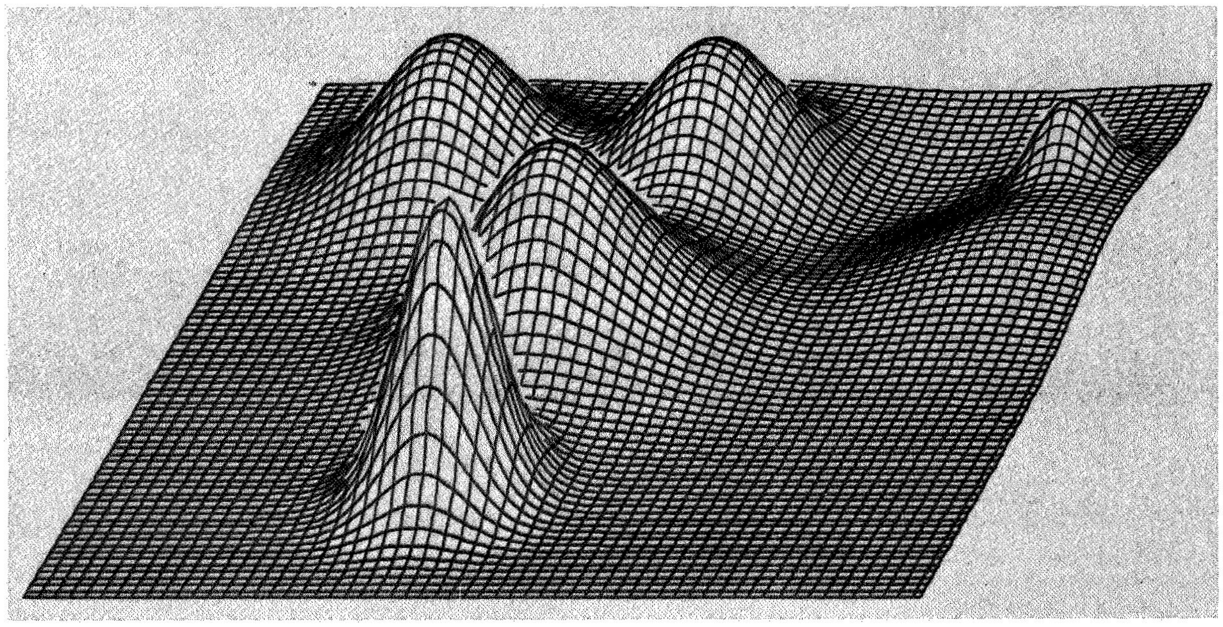


Fig. 3. Perspective drawing of contour configuration

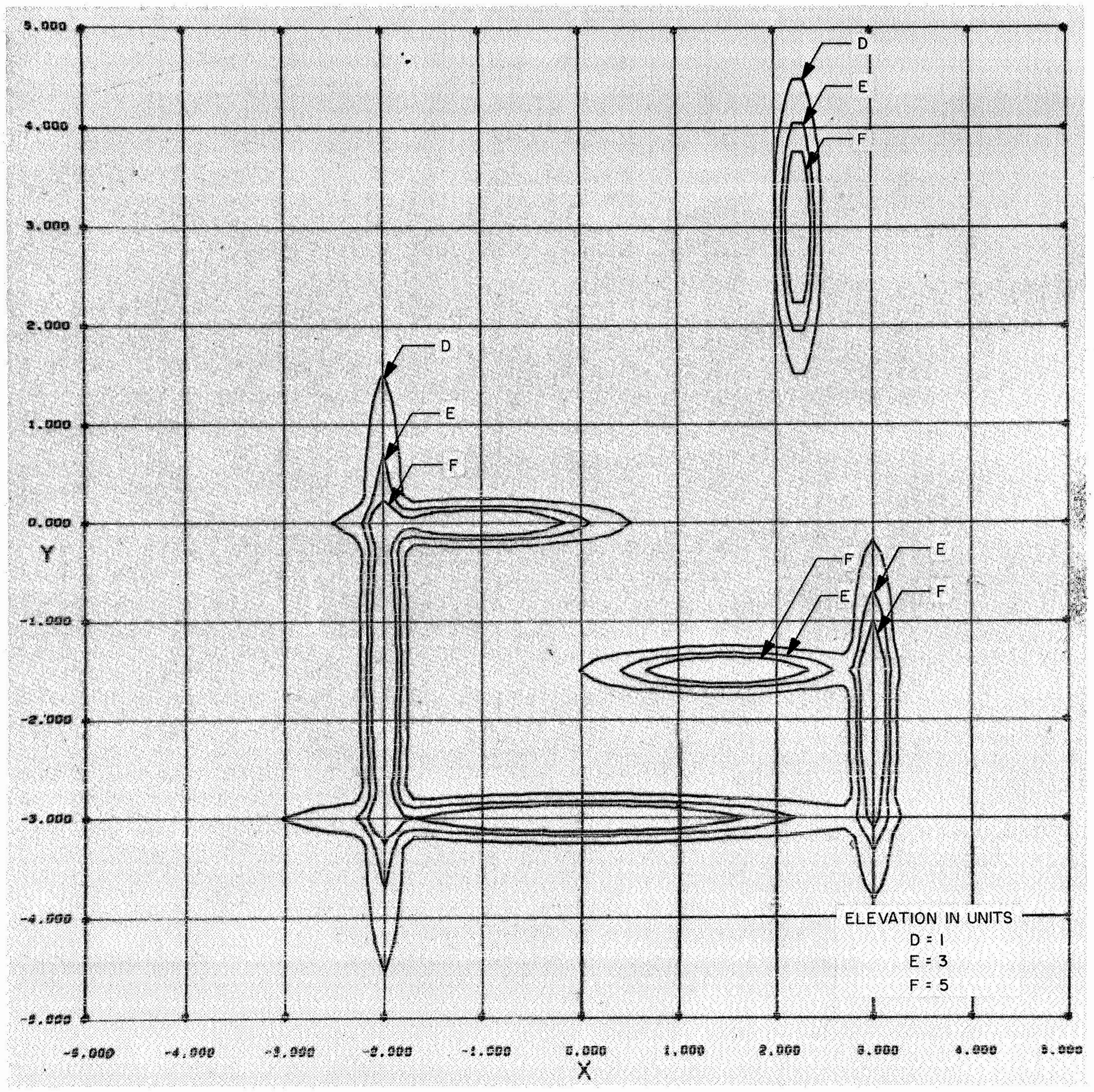


Fig. 4. A maze constructed by use of Gaussian density functions

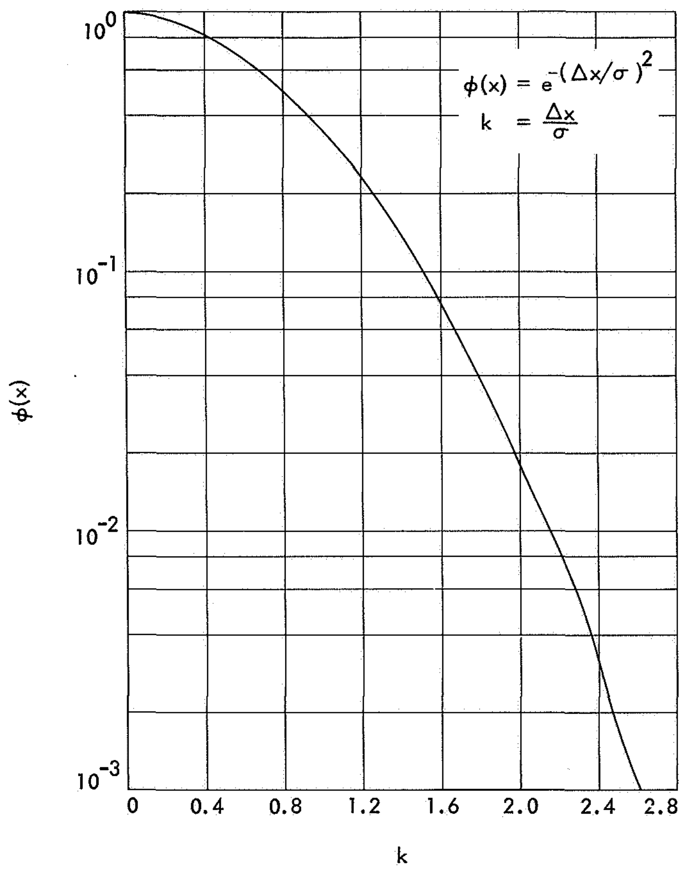
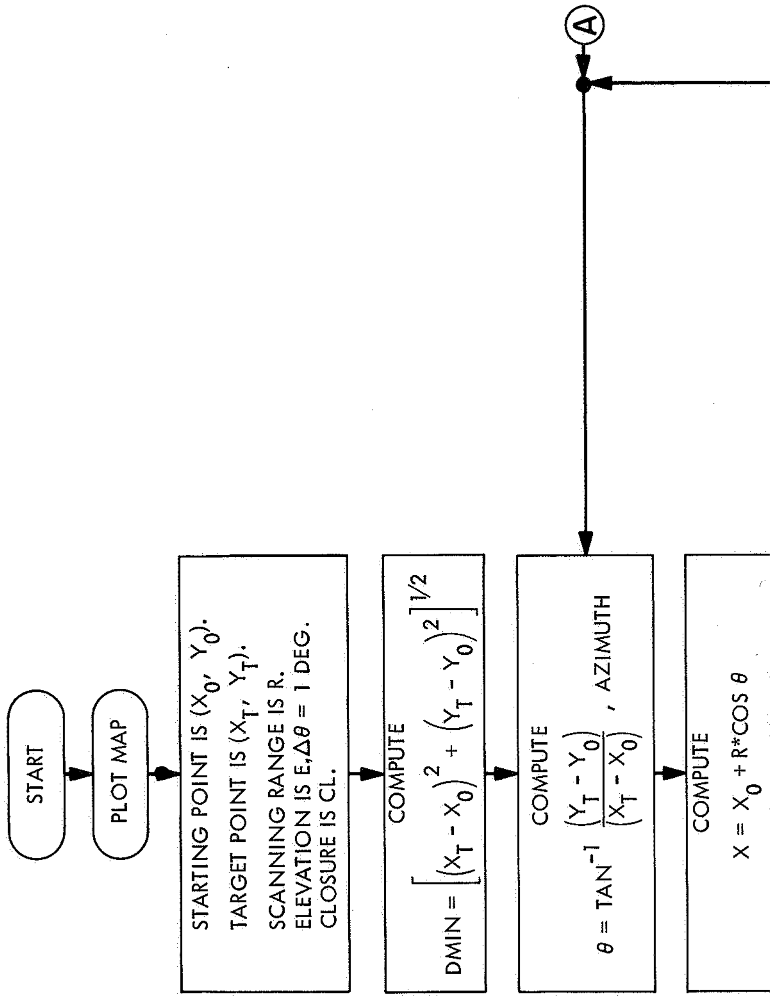


Fig. 5. Graph for calculating and constructing any configuration required for the simulation of terrains


A
START
PLOT MAP
STARTING POINT IS (X_0, Y_0).
TARGET POINT IS (X_T, Y_T).
SCANNING RANGE IS R.
ELEVATION IS E, $\Delta\theta$ = 1 DEG.
CLOSURE IS CL.
COMPUTE
DMIN = $\left[(X_T - X_0)^2 + (Y_T - Y_0)^2\right]^{1/2}$
COMPUTE
$\theta = \mathrm{TAN}^{-1} \frac{(Y_T - Y_0)}{(X_T - X_0)}$, AZIMUTH
COMPUTE
$X = X_0 + R^*\mathrm{COS}\,\theta$

FOLDOUT FRAME /

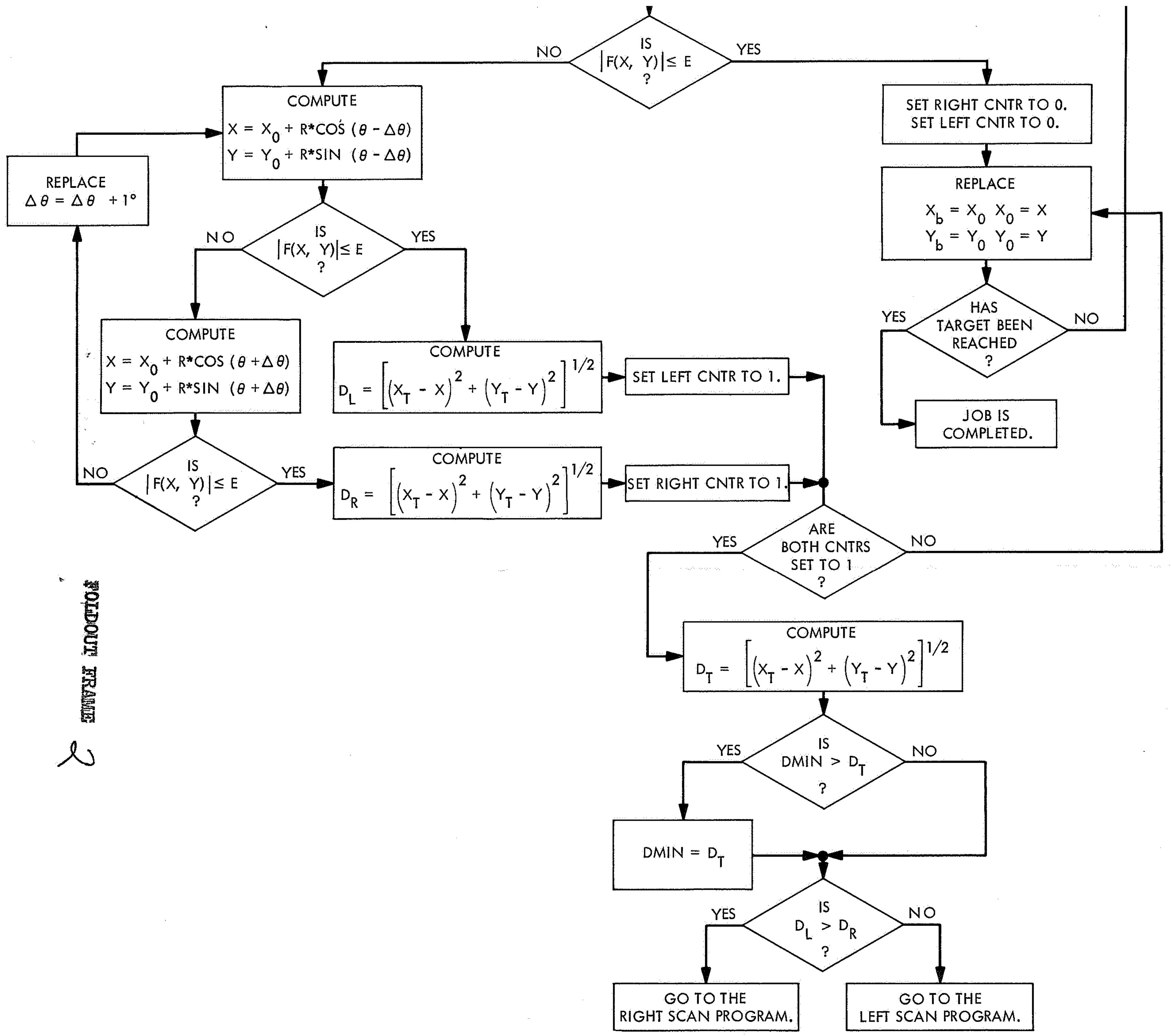

IS |F(X, Y)|≤ E ?
NO
YES
COMPUTE X = X0 + R*COS (θ - Δθ) Y = Y0 + R*SIN (θ - Δθ)
REPLACE Δθ = Δθ + 1°
IS |F(X, Y)|≤ E ?
NO
YES
COMPUTE X = X0 + R*COS (θ + Δθ) Y = Y0 + R*SIN (θ + Δθ)
COMPUTE D_L = [(X_T - X)^2 + (Y_T - Y)^2]^1/2
IS |F(X, Y)|≤ E ?
NO
YES
COMPUTE D_R = [(X_T - X)^2 + (Y_T - Y)^2]^1/2
SET LEFT CNTR TO 1.
SET RIGHT CNTR TO 1.
ARE BOTH CNTRS SET TO 1 ?
YES
NO
COMPUTE D_T = [(X_T - X)^2 + (Y_T - Y)^2]^1/2
IS DMIN > D_T ?
YES
NO
DMIN = D_T
IS D_L > D_R ?
YES
NO
GO TO THE RIGHT SCAN PROGRAM.
GO TO THE LEFT SCAN PROGRAM.
SET RIGHT CNTR TO 0. SET LEFT CNTR TO 0.
REPLACE X_b = X_0 X_0 = X Y_b = Y_0 Y_0 = Y
HAS TARGET BEEN REACHED ?
YES
NO
JOB IS COMPLETED.

Fig. 6. Flow chart of computer program, main algorithm

FOLDOUT FRAME 2

Page intentionally left blank

PRECEDING PAGE BLANK NOT FILMED.

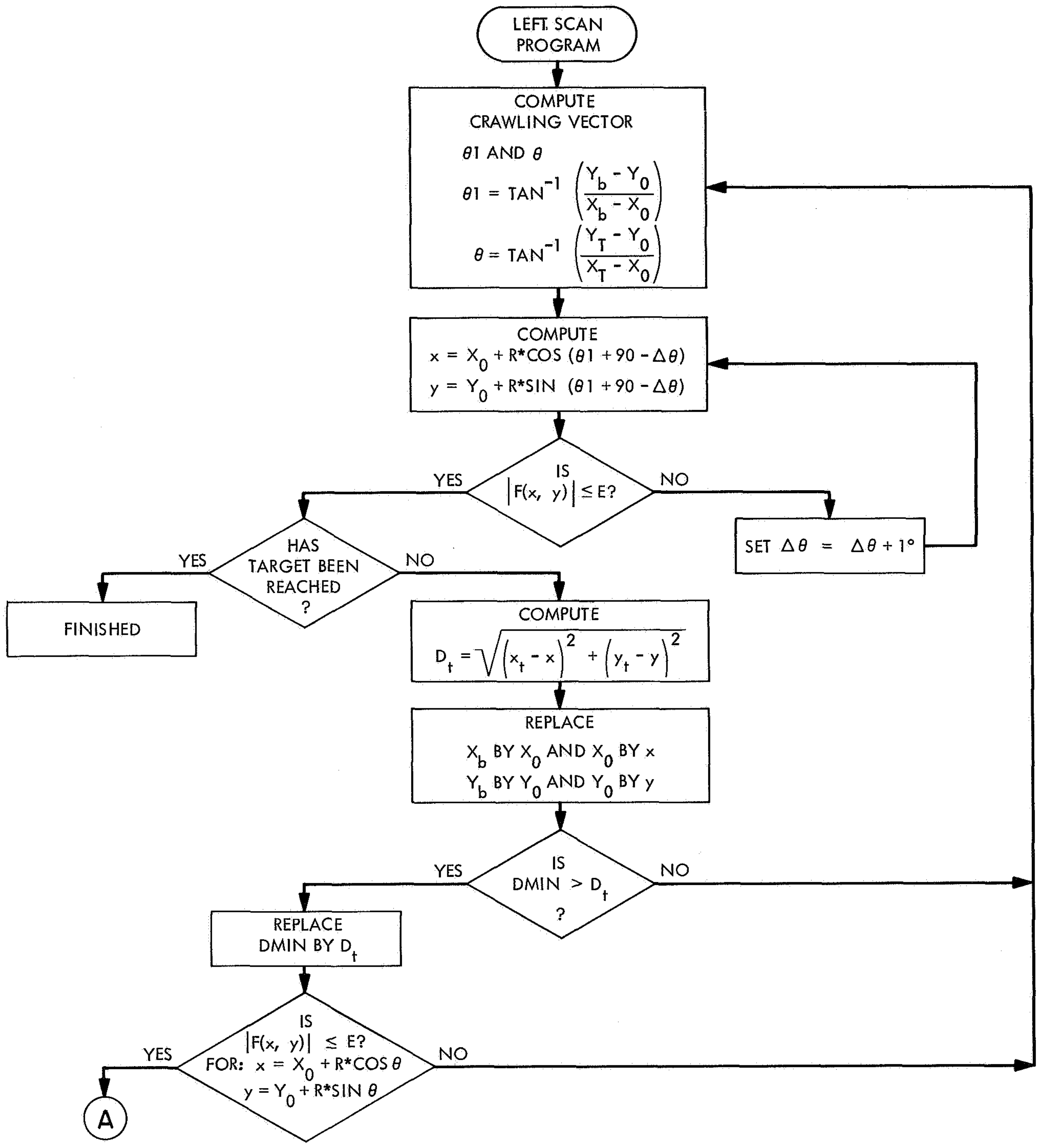

LEFT SCAN PROGRAM
COMPUTE CRAWLING VECTOR
$\theta 1$ AND θ
$\theta 1 = \mathrm{TAN}^{-1}\left(\frac{Y_b - Y_0}{X_b - X_0}\right)$
$\theta = \mathrm{TAN}^{-1}\left(\frac{Y_T - Y_0}{X_T - X_0}\right)$
COMPUTE
$x = X_0 + R*\mathrm{COS}(\theta 1 + 90 - \Delta\theta)$
$y = Y_0 + R*\mathrm{SIN}(\theta 1 + 90 - \Delta\theta)$
IS $|F(x, y)| \leq E$?
YES
NO
SET $\Delta\theta = \Delta\theta + 1°$
HAS TARGET BEEN REACHED ?
YES
NO
FINISHED
COMPUTE
$D_t = \sqrt{(x_t - x)^2 + (y_t - y)^2}$
REPLACE
X_b BY X_0 AND X_0 BY x
Y_b BY Y_0 AND Y_0 BY y
IS DMIN > D_t ?
YES
NO
REPLACE DMIN BY D_t
IS $|F(x, y)| \leq E$?
FOR: $x = X_0 + R*\mathrm{COS}\,\theta$
$y = Y_0 + R*\mathrm{SIN}\,\theta$
YES
NO
A

Fig. 7. Flow chart of computer program, left scan algorithm

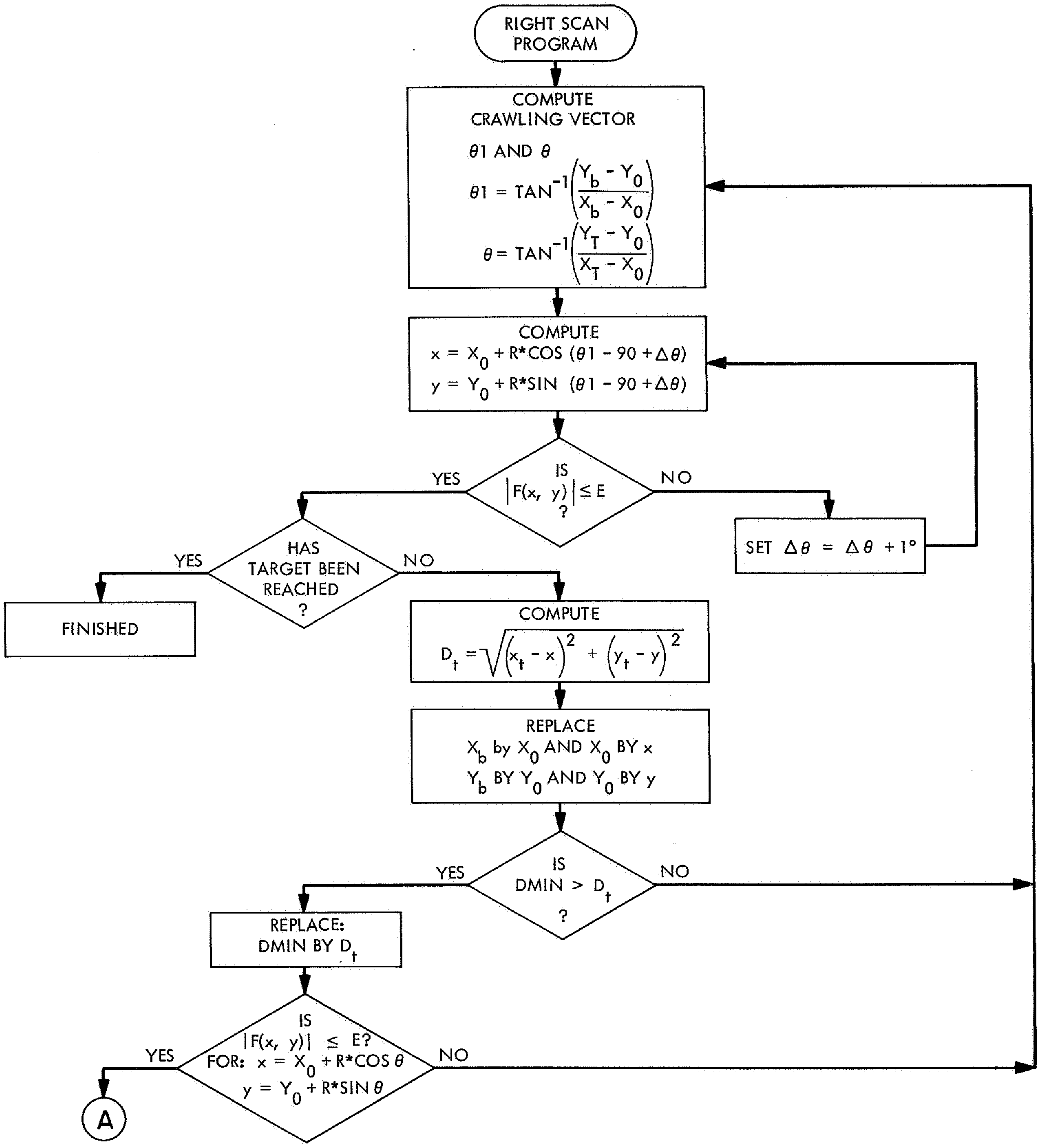

RIGHT SCAN PROGRAM
COMPUTE CRAWLING VECTOR
$\theta 1$ AND θ
$\theta 1 = \mathrm{TAN}^{-1}\left(\frac{Y_b - Y_0}{X_b - X_0}\right)$
$\theta = \mathrm{TAN}^{-1}\left(\frac{Y_T - Y_0}{X_T - X_0}\right)$
COMPUTE
$x = X_0 + R*\mathrm{COS}\ (\theta 1 - 90 + \Delta\theta)$
$y = Y_0 + R*\mathrm{SIN}\ (\theta 1 - 90 + \Delta\theta)$
IS $|F(x, y)| \leq E$?
YES
NO
SET $\Delta\theta = \Delta\theta + 1°$
HAS TARGET BEEN REACHED ?
YES
NO
FINISHED
COMPUTE
$D_t = \sqrt{(x_t - x)^2 + (y_t - y)^2}$
REPLACE
X_b by X_0 AND X_0 BY x
Y_b BY Y_0 AND Y_0 BY y
IS DMIN > D_t ?
YES
NO
REPLACE: DMIN BY D_t
IS $|F(x, y)| \leq E$?
FOR: $x = X_0 + R*\mathrm{COS}\ \theta$
$y = Y_0 + R*\mathrm{SIN}\ \theta$
YES
NO
A

Fig. 8. Flow chart of computer program, right scan algorithm

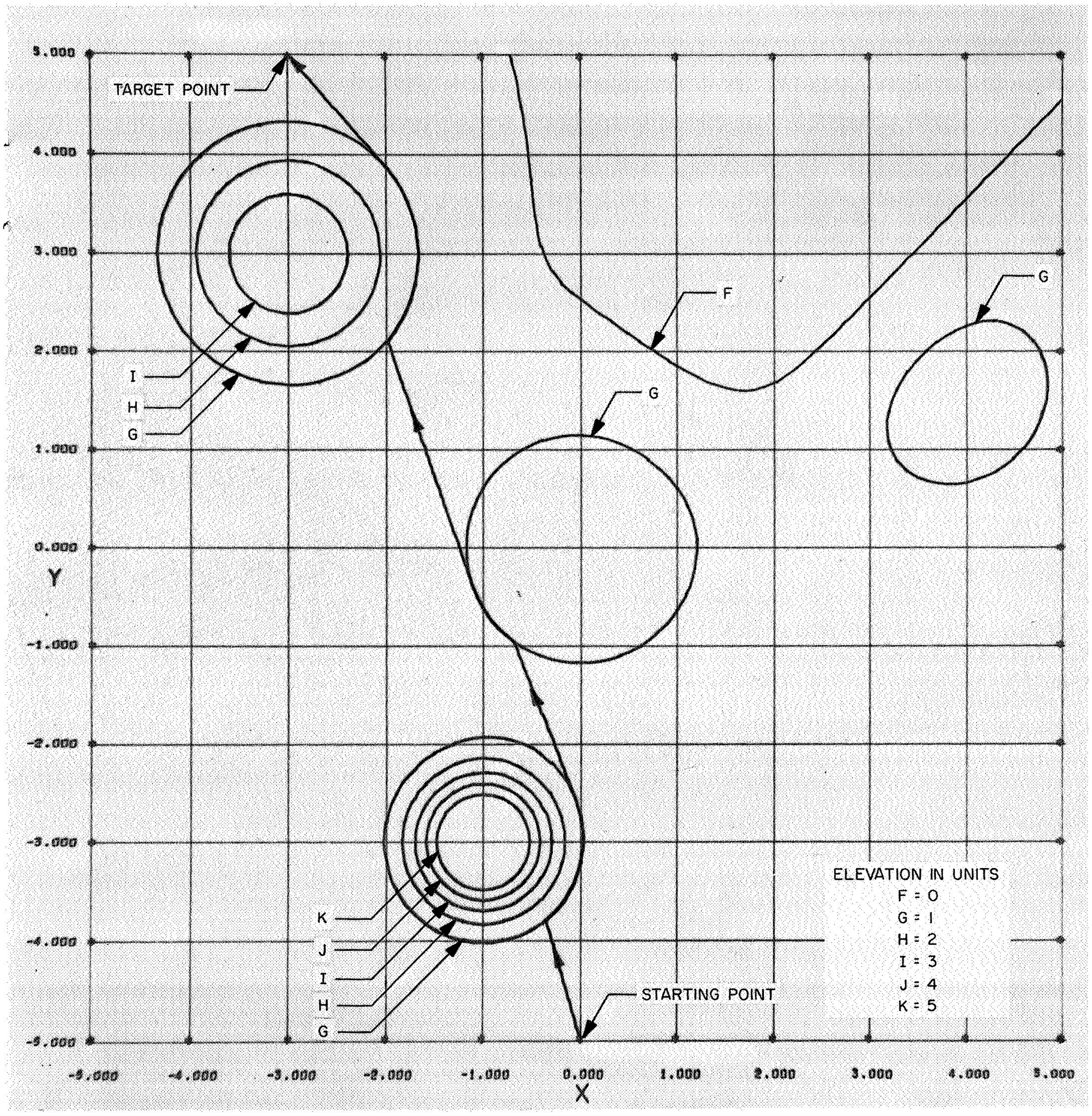


Fig. 9. A simple obstacle course of five Gaussian hills

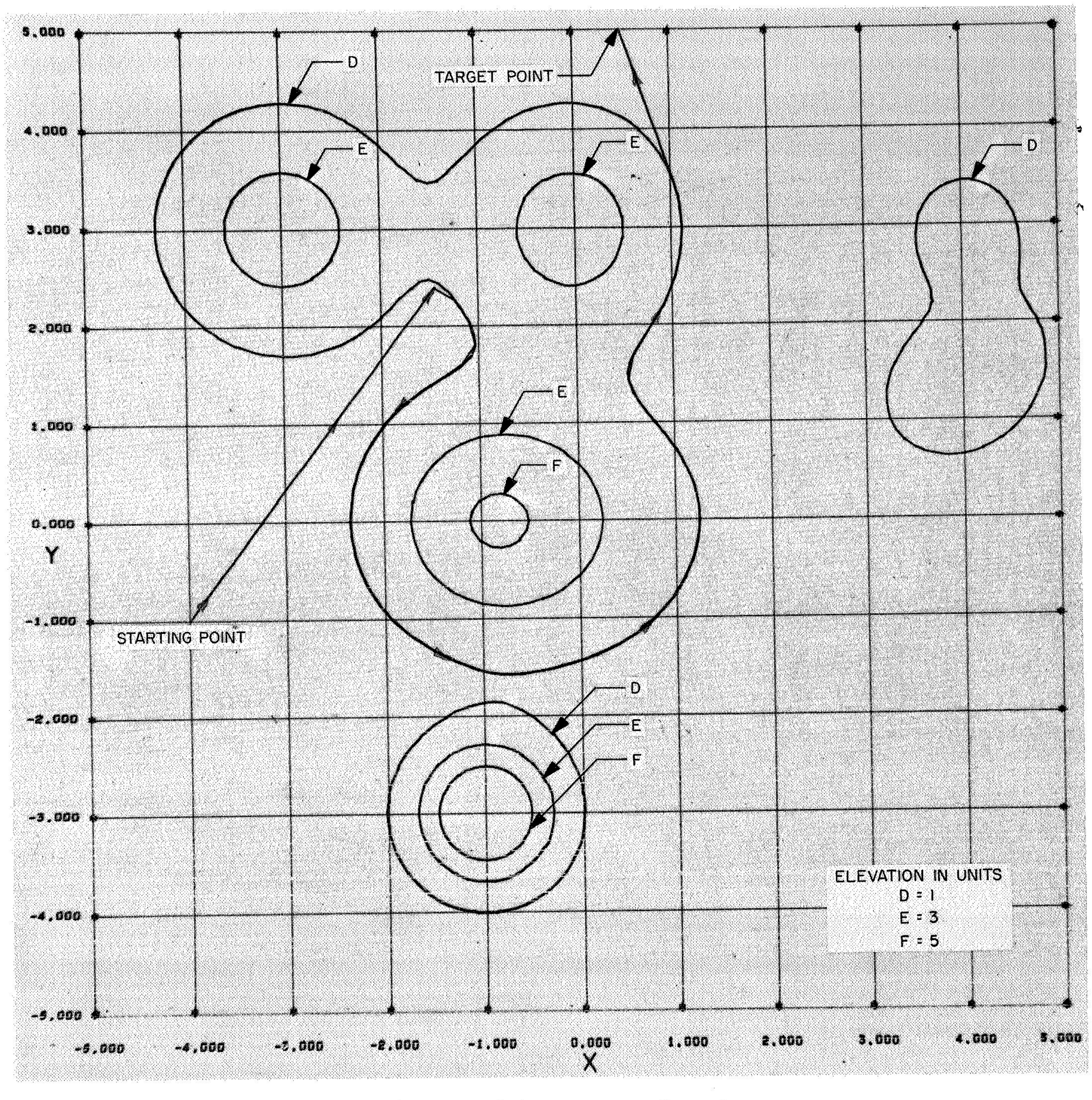


Fig. 10. A simple box canyon configuration

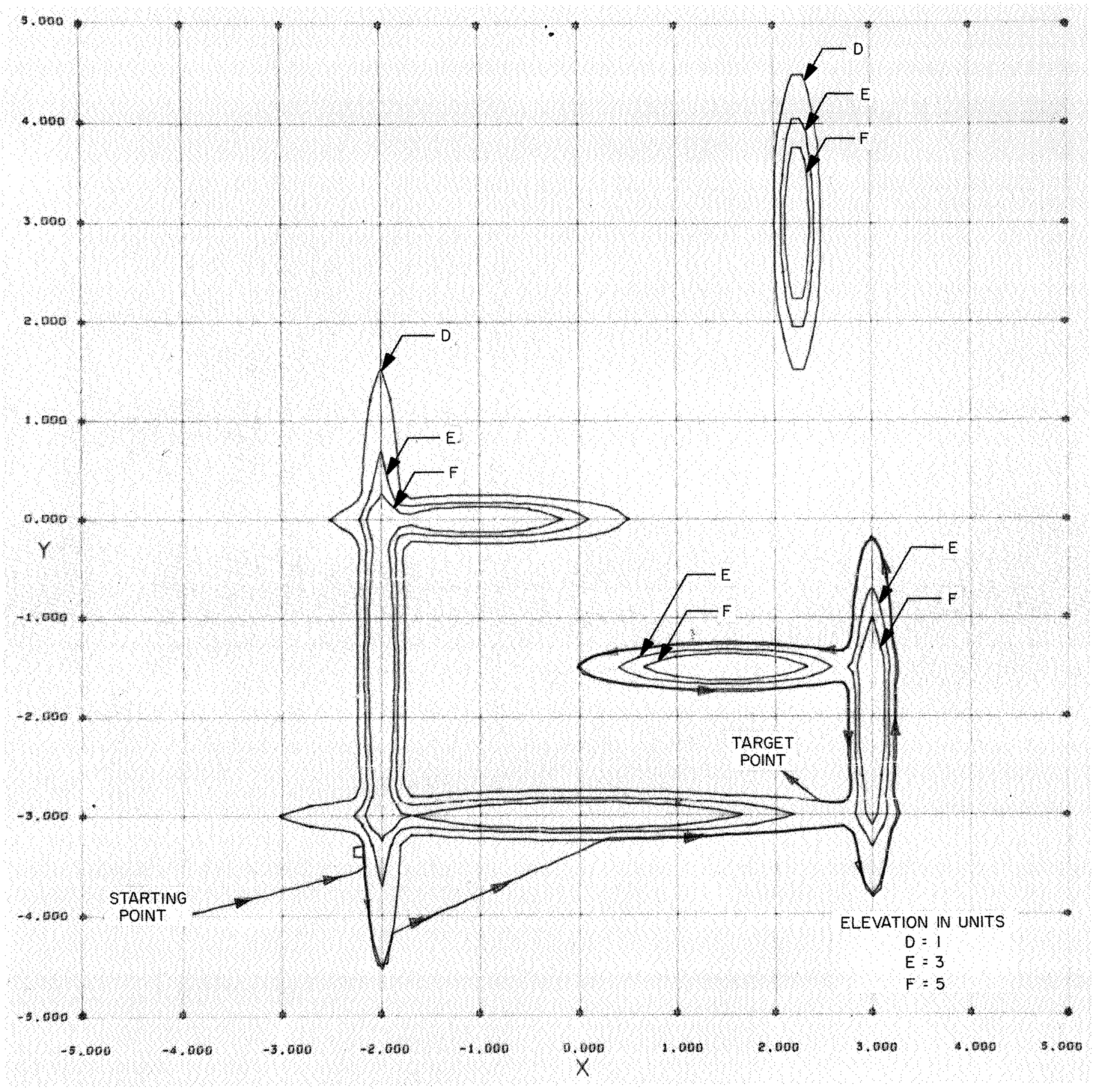


Fig. 11. A maze in which the starting point is outside of the maze and the target point is inside

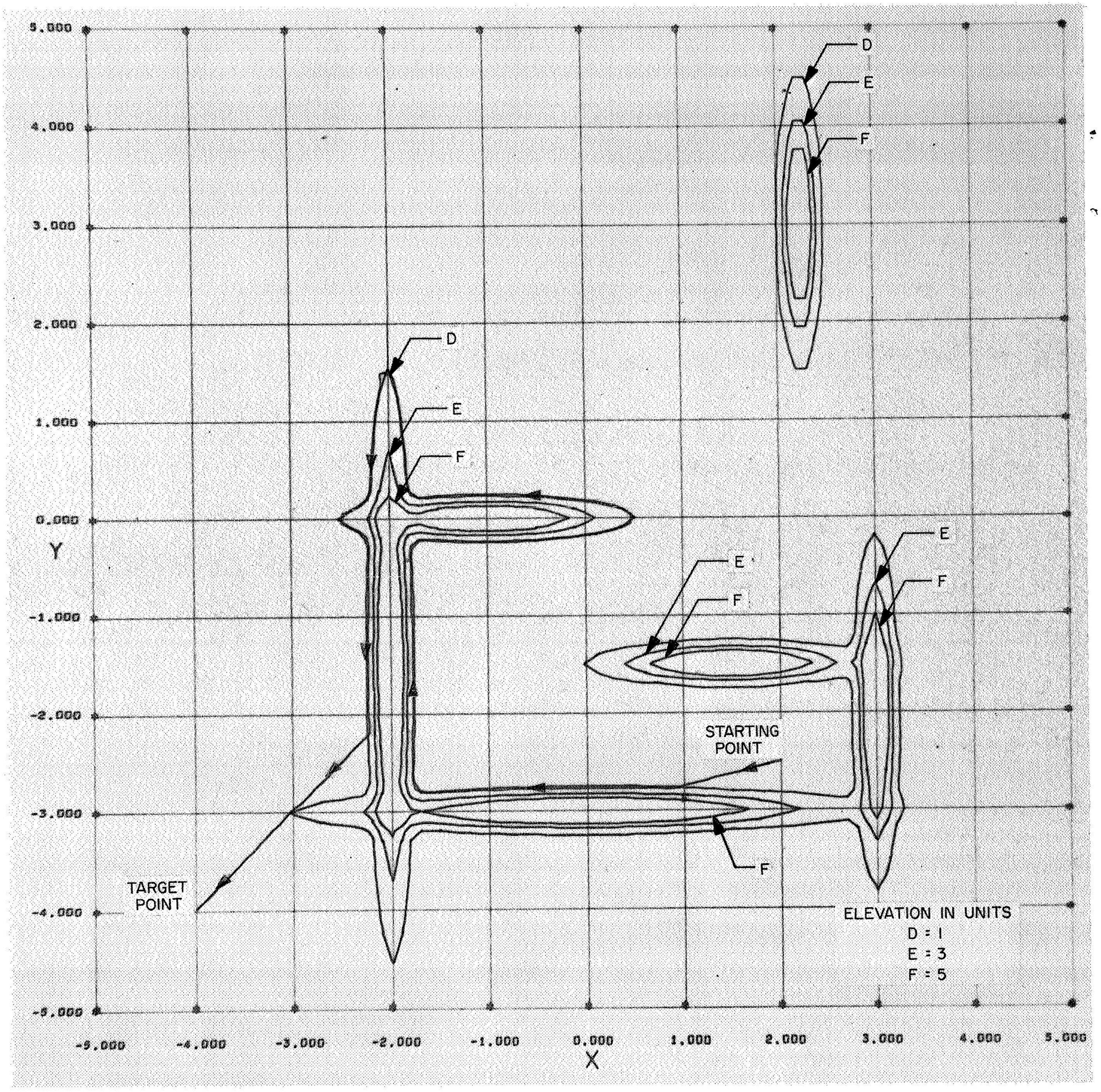


Fig. 12. Same maze in which the starting point is inside the maze and the target point is outside

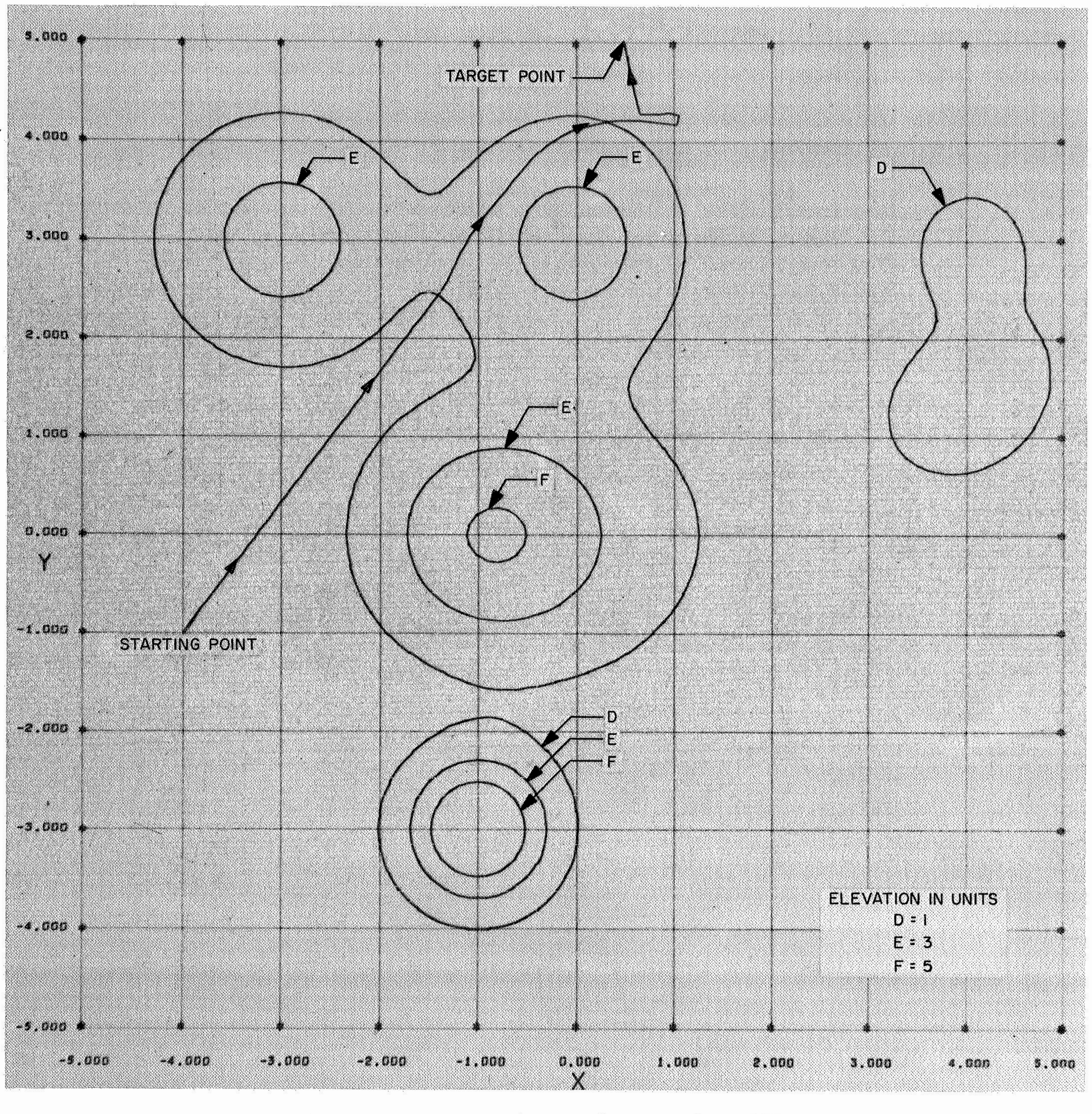


Fig. 13. Result of the robot crawling on and over hills

References

1. Moore, E. F., "Shortest Path through a Maze," *Annals of the Computation Laboratory of Harvard University*, Vol. 30, pp. 285–292, Harvard University Press, Cambridge, Mass., 1959.

2. Lee, C. Y., "An Algorithm for Path Connections and Its Applications," *IEEE Trans. Electron. Comput.*, pp. 346–365, Sept. 1961.

3. Miller, B. P., et al., *Roving Vehicle Motion Control*, Final Report TR 67-60. AC Electronics, Defense Research Laboratories, Santa Barbara, Calif., Dec. 1967.

4. Lawson, C. L., Block, N., and Garrett, R. S., "Computer Subroutine for Contour Plotting," in *Supporting Research and Advanced Development*, Space Programs Summary 37-32, Vol. IV, p. 18. Jet Propulsion Laboratory, Pasadena, Calif., Feb. 1–Mar. 31, 1965.

5. Freeman, H., and Loutrel, P. P., "An Algorithm for the Solution of the Two-Dimensional 'Hidden-Line' Problem," *IEEE Trans. Electron. Comput.*, Vol. EC-16, No. 6, pp. 784–790, Dec. 1967.