AN ALGORITHM FOR COMPUTER GUIDANCE OF A MANIPULATOR IN BETWEEN OBSTACLES

By

LUIS ANGEL LOEFF

Ingeniero Industrial

Universidad de la Republica

Montevideo, Uruguay

1971

Submitted to the Faculty of the Graduate College
of the Oklahoma State University
in partial fulfillment of the requirements
for the Degree of
MASTER OF SCIENCE
December, 1973

AN ALGORITHM FOR COMPUTER GUIDANCE OF A MANIPULATOR IN BETWEEN OBSTACLES

Thesis Approved:

Thesis Adviser

Dean of the Graduate College

ACKNOWLEDGEMENTS

I want to express my appreciation to my advicer Dr. A. H. Soni for his encouragement and cooperation.

I would also like to thank my friends Amnon Vadasz, Dilip Kohli, Jack Lee, Louis Torfason and Arun Chaudhari for their understanding and help.

I wish to acknowledge the National Science Foundation grant GK-21029 which made part of this study possible.

Finally my deep appreciation to my parents and uncle for their moral and material support.

...

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

A manipulator is a device that can perform certain tasks and has definite similarity to the human arm and hand. Originally manipulators were used to substitute for the human arm and hand in hostile environments, but with the advance in technology, use of manipulators extended far beyond hostile environments to simple, tedious and hazardous tasks. The control of manipulators is a problem by itself and only recently have there been breakthroughs in this particular area of technology.

Manipulator control can be divided into two broad classes, human control and computer control. In human control, the manipulator is coupled to the human arm in such a way that both movements are highly correlated. In the computer control, it is the job of the digital computer to process the information related to the manipulator's task and the obstacles to be avoided. Computer control can be further subdivided according to the way the information regarding the manipulator's task and the obstacles is fed into the computer. If the information regarding the task is all contained in the memory of the computer before the manipulator starts moving and no further information enters the computer during the realization of the task, then we have open-loop control. If the information related to the manipulator's positions and tasks and its obstacles is fed during the realization of the

task as for example, with an electronic sensory device, then we have a closed-loop control. This thesis proposes a method by which a manipulator is guided through obstacles using both open-loop and closed-loop control features. The description of the obstacles is stored in the memory of the digital computer before the task starts, but the position of the manipulator is fed concurrently with the realization of the task.

In Chapter II we will develop the algorithm which will fulfill the proposed goal.

In Chapter III we will present results and analysis.

In Chapter IV we present conclusions and some suggestions for future work.

Chapter V contains computer programs and instructions on how to use the algorithm.

In the next section we present a brief history of manipulators and related fields.

History of Manipulators

The development of manipulators started with the development of atomic energy. In order to protect an operator from radioactive materials, equipment was needed that could duplicate the operator's capabilities without exposing him to a hazardous environment. Research on general purpose manipulators was started in Argonne National Laboratory in 1947. The first manipulator built at Argonne had six degrees of freedom, were controlled by mechanical drives and had hydraulically operated grips. Later, electric drives were substituted for mechanical drives. They were suitable for simple tasks, but because they

lacked force feedback then although an operator exerted a light pressure on the grip, the manipulator reacted always with the same force making it unsuitable for handling delicate tasks; as can be deduced from the above, the early manipulators fell into the classification as human controlled. Later on, in 1948 the people at Argonne decided to develop manipulators with force feedback and with motion very similar to the human hand. These are called master-slave manipulators because both motions are coupled in such a way that the motion of the slave follows the motion of the master and the force in the slave is reflected attenuated at the master. Several of these were built and they are still commercially available today.

These manipulators have several drawbacks. They require the operator and the manipulator to be physically close together. Also the strength of the manipulator is limited by the strength of the operator. Later on GEC and Argonne developed other manipulators that could be controlled from far away through a cable. However, because of their higher cost, their use is limited. Powered manipulators, not of the master-slave types, have been developed by AMF, GEC, Westinghouse, FMC, etc. The most advanced human controlled type of manipulator is built by GEC under the name CAMS (Cybernetic Anthropomorphous Machine Systems). All these manipulators, being human controlled, suffer from the inherent limitations of human beings; they are subject to fatigue and are error prone. Efforts are being made at present to create systems in which man has no more than a supervisory task.

Computer controlled manipulators offer a partial solution to human limitations. In space exploration for example, the delay involved between a human operator on earth and a manipulator in some remote planet

points out the necessity of controlling a manipulator with a computer. Computer-manipulator systems are now being built and used in material handling applications such as AMF's Vesatran and Unimation Inc.'s Unimate. These machines are of the open-loop type and are programmed through a predetermined series of positions. They are used to perform specific operations having a fixed cycle. They cannot make decisions. However if the parts are not in the right positions, the machine will not operate successfully. They must be reprogrammed each time the task varies. It is desirable therefore to include some decision-making capability to the manipulator control system.

Research on systems with the above mentioned capabilities has been and is being conducted mainly at two universities, M. I. T. and Stanford. At M. I. T., Ernst [1] using a manipulator with sensory feedback, developed a system capable of stacking blocks. Ernst's work was further extended to include visual inputs and capability of manipulating objects. Future goals include the development of a completely autonomous system capable of manipulating objects with a sophisticated level of decision making ability. At Stanford, Rosen [2] and others developed a computer controlled mobile vehicle that performs tasks on a real environment. There, most of the emphasis is placed on developing the feedback loop of the system (reception of visual and other sensory information to direct the system-vehicle in this case towards the completion of tasks requiring the abilities to plan ahead and learn from previous experience). At other universities [3], similar research has been conducted in an effort to increase the decision making capabilities of manipulator systems.

Contribution of the Proposed Research

The algorithm developed represents an alternative to the few already in existence for achieving the same goal. In addition the method can be used with little variation for virtually unlimited types of arms and obstacles. It does not require the complicated visual sensors with very sophisticated pattern recognition software. It represents an advance with respect to the method outlined by Pieper in his doctoral dissertation because it does not need to solve equations for the angles that locate the hand at a certain point. It does this automatically as the arm is advancing towards its objective. Future possibilities of this method include the addition of logical decision capabilities in order to select the best way to reach its objective. This is accomplished after the criteria of what "best" means has been adopted. The next chapter will explain the derivation of the algorithm.

CHAPTER II

ALGORITHM DEVELOPMENT

Selection of the Manipulator Configuration

The primary criterion which was considered when selecting a configuration was generality. The configuration needed to be general enough as for representing most, if not all existing real manipulators. The approach followed was to start with a generalized link, with several degrees of freedom. Then, any real manipulator can be represented with a certain number of generalized links, each of them with none, one or more degrees of freedom of motion frozen. The generalized link is shown in Figure 1. Let this link be the ith link of an open kinematic chain. The end points of this link are P_{i-1} and P_i. Each link contains a cylinder pair collinear with the link itself and a revolute pair perpendicular to the link at the point P_i. The extension of the cylinder pair will be called a_i and its rotation will be called α_i. The rotation at the revolute pair will be denoted θ_i. In our analysis we will consider that the hand is small enough not to influence the algorithm. We will assume therefore, that as long as the end point of the manipulator, in which the hand is attached keeps outside a certain distance from the obstacles, then it could be assured that the hand does not touch any obstacle. Let's now use the generalized link to

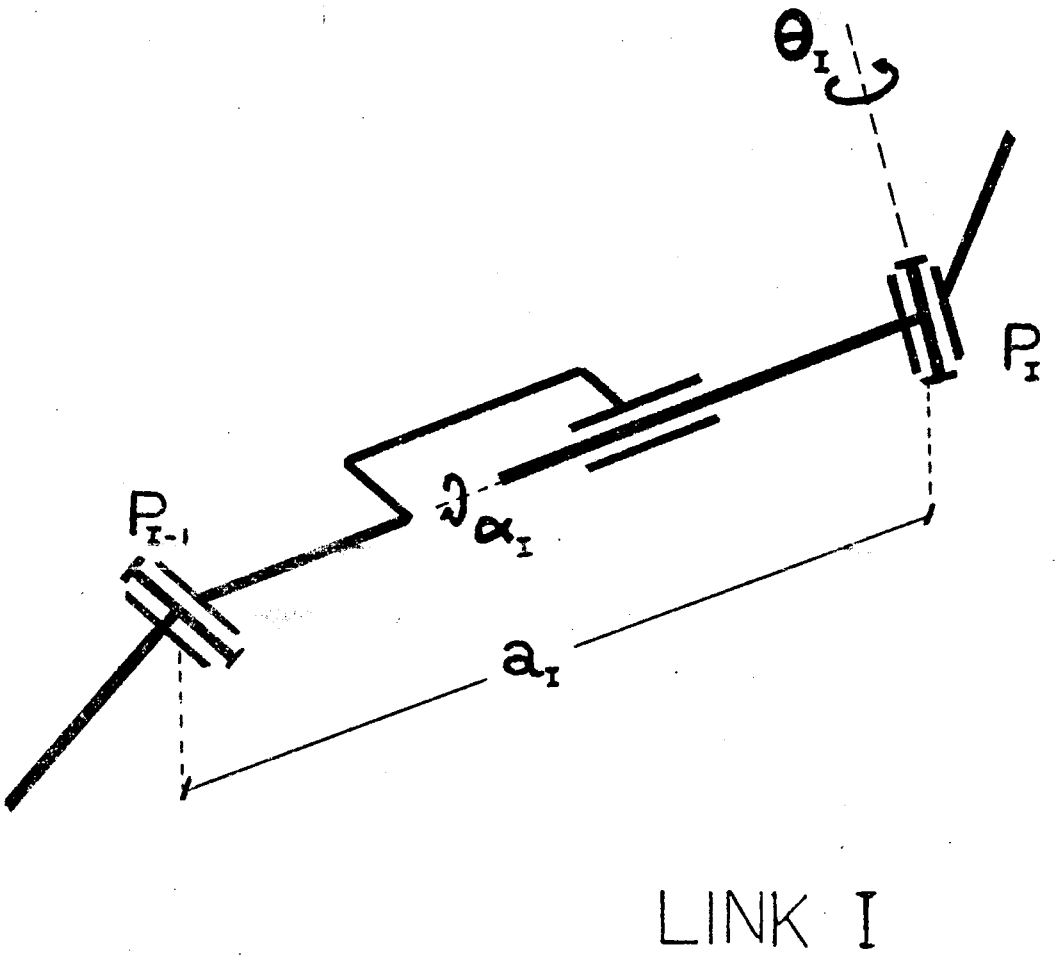

Figure 1. Generalized Link

describe some configuration of commercial manipulators. In the ith link if some parameter is fixed, we will denote it by equating the parameter to its value when the parameter is a rotation or by stating that the increment of the parameter is zero when it is a length. So for example if in a manipulator, α of link 1 is fixed and equal to 0°, θ of link 2 is fixed and equal to 90° and if a of link 3 is fixed, then we will denote the manipulator by ($\alpha_1 = 0^\circ$, $\theta_2 = 90^\circ$, $\Delta a_3 = 0$, 5) where 5 denotes number of links. Before going to specific examples it is necessary to define the number of links of a manipulator. We will assume a definition based on convenience and practicality that will have its advantages in implementing the algorithm. This number consists of all the links that do not have either of their ends clamped to the ground. It might be argued that when the link clamped to the ground stretches, it should be considered as a link, but as in most practical cases this link does not stretch, the definition is justified. In order to use an array of numbers in the computer the indexes of the array cannot be zero, so the link that has an end clamped to the ground will be assigned number 1. Consider the practical manipulator structures shown in Figures 2 and 3. A simple analysis concludes that the structures have descriptions ($a_1 = 0$, $\alpha_1 = 0^\circ$, $\Delta a_2 = 0$, $\alpha_2 = 90^\circ$, $\Delta a_3 = 0$, $\alpha_3 = 0^\circ$, $\Delta a_4 = 0$, $\Delta a_5 = 0$, 4) and ($\Delta a_1 = 0$, $\alpha_1 = 0^\circ$, $\alpha_2 = 90^\circ$, $\Delta a_2 = 0$, $\Delta a_4 = 0$, 3) respectively. Having seen that the generalized links can be used to represent practical manipulators, it is time to present the foundations of the algorithm.

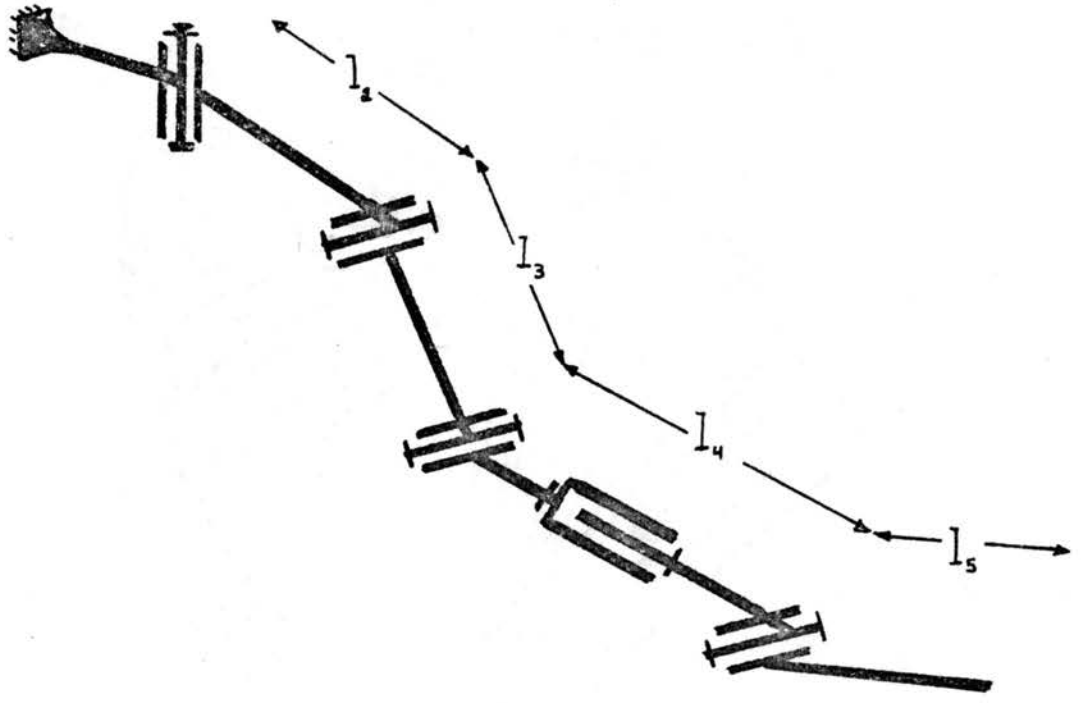

Figure 2. Humanoid Type Manipulator

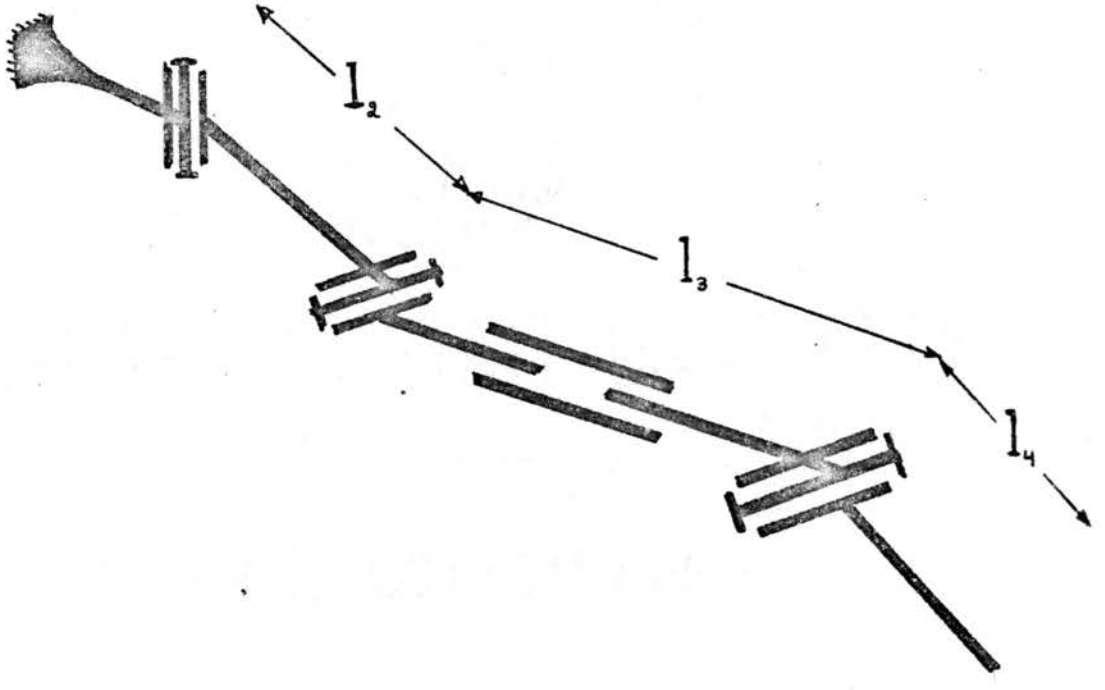

Figure 3. Unimate Type Manipulator

Mathematical Description of the Manipulator

Any manipulator formed by a chain of N generalized links will have at most $3(N+1)$ parameters although α_{N+1} and θ_{N+1} will not really be considered because they deal with the position of the hand. In order to describe the configuration of the manipulator we need to specify each of the $3N+1$ parameters which are (a_1, a_2,....., a_{N+1}, α_1, α_2,..., α_N, θ_1, θ_2,...., θ_N). Describing the configuration means in our case knowing the positions with respect to a fixed frame of the end points of all links. Now let's discuss what we mean by a guidance algorithm. Given a configuration as indicated previously by (a_1,..... ..., θ_N) , if we can find an incremental configuration (Δa_1,..., $\Delta\theta_N$) as a function of the configuration (a_1,...., θ_N), then by successive iterations (compute the incremental configuration and add it to the configuration to obtain a new configuration, and so on), we can move the manipulator in a certain way. Let's now consider t to be an independent variable. (It is not important whether t is time or not, because we are interested mainly in trajectory generation and not in solving dynamic problems). Then $\{\Delta a_1, \ldots, \Delta\theta_N\} = \{\dot{a}_1, \ldots, \dot{\theta}_N\}\Delta t$ where the dots indicate the derivative with respect to the parameter t. Then for each (a_1,..., θ_N) we should find ($\dot{a}_1$,..., $\dot{\theta}_N$). Let's denote the end points of the links of the manipulator by P_1, P_2,....., P_i,..., P_{N+1} where P_i represents the point of coordinates (x_i, y_i, z_i) and its derivatives with respect to the independent variable t by $\dot{P}_1$, ..., $\dot{P}_i$,..., $\dot{P}_{N+1}$ where $\dot{P}_i$ represents ($\dot{x}_i$, $\dot{y}_i$, $\dot{z}_i$). Our algorithm has three clearly differenciated parts:

1) Given ($a_1,\ldots, \theta_N$) compute ($P_1,\ldots., P_{N+1}$).

2) Knowing ($P_1,\ldots., P_{N+1}$), the destination of P_{N+1}, and the description of the obstacles, generate a set of "velocities" or influences (either of these two tems will be used indistinctly) ($\dot{P}_1,\ldots, \dot{P}_{N+1}$) that will be compatible with the physical dimensions of the links in the manipulator.

3) Once ($\dot{P}_1,\ldots, \dot{P}_{N+1}$) are computed calculate ($\dot{a}_1,\ldots, \dot{\theta}_N$).

Steps 1) and 3) are straightforward although their derivation is rather lengthy. Step 2) is not straightforward and its solution requires engineering ingenuity. The analytical tool to be used in steps 1) and 3) are the 4x4 matrices proposed by Hartenberg and Denavit [4]. This matrix is to be denoted by $[R_i]$ enclosed by square brackets, where i denotes the link number to which the matrix is referred. The positions of the end points of the links in the manipulator will be denoted by $P_{i,j}$ where i is the number of the end point as defined previously, and j indicated the link number to which a frame of reference is attached. Its form will be:

$$P_{i,j} = \begin{Bmatrix} x_i \\ y_i \\ z_i \end{Bmatrix}_j \quad \text{or} \quad \begin{Bmatrix} 1 \\ x_i \\ y_i \\ z_i \end{Bmatrix}_j \qquad (2\text{-}1)$$

The second form is the one that will be used in our derivation because we are using 4x4 matrices.

Position Description

Our goal is to know the location of all points ($P_1,\ldots, P_{N+1}$) with respect to a fixed frame of reference attached to link 1. To do this we fix on each link i a frame of reference i whose origin coincides with the end point P_{i-1}, whose x axis coincides with the line defined by points P_{i-1} and P_i, and whose z axis coincides with the axis of the revolute pair at P_{i-1}. Let's now consider a generic link i as indicated in Figure 4 with its reference frame attached to it. Suppose that we know the coordinates of a certain point with respect to frame i+1, and we want to find the position of the same point with respest to frame i. Hartenberg and Denavit discovered a matrix that accomplishes this coordinate transformation. Our matrix will differ from theirs by the order in which the rotations α and θ are taken. To transform a certain point P_g referred to frame i+1 into frame $i+\frac{1}{2}$, we have to multiply $P_{g,i+1}$ by the matrix:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta_i & -\sin\theta_i & 0 \\ 0 & \sin\theta_i & \cos\theta_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (2\text{-}2)$$

This matrix will be denoted by $[R_{\theta_i}]$. To find the coordinates of point P_g given in frame $i+\frac{1}{2}$ to frame i, we have to multiply $P_{g,i+\frac{1}{2}}$ by the matrix:

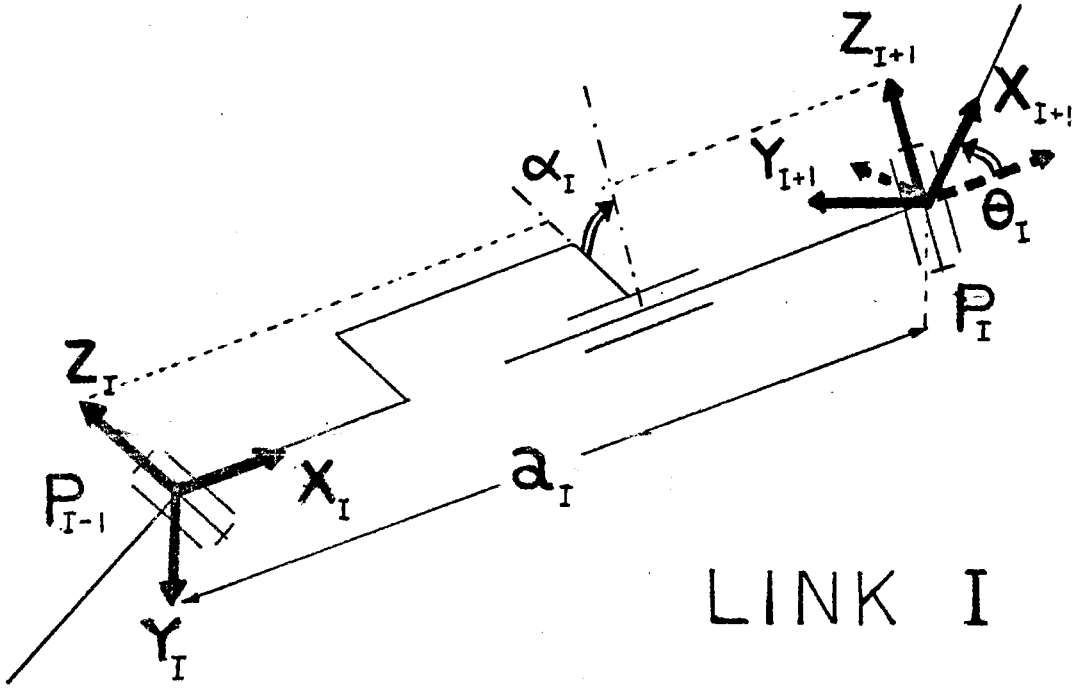

Figure 4. Frames of Reference

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ a_i & 1 & 0 & 0 \\ 0 & 0 & \cos\alpha_i & -\sin\alpha_i \\ 0 & 0 & \sin\alpha_i & \cos\alpha_i \end{bmatrix} \tag{2-3}$$

This matrix will be denoted by $[R_{\alpha_i}]$. Then, $P_{g,i}$ as a function of $P_{g,i+1}$ will be:

$$P_{g,i} = [R_{\alpha_i}][R_{\theta_i}]P_{g,i+1}$$

Or, after multiplying the matrices in the indicated order:

$$P_{g,i} = [R_i]P_{g,i+1}$$

Equation (2-4) is the abbreviated form of (2-5) shown below,

$$\begin{Bmatrix} 1 \\ x_g \\ y_g \\ z_g \end{Bmatrix}_i = \begin{bmatrix} 1 & 0 & 0 & 0 \\ a_i & \cos\theta_i & -\sin\theta_i & 0 \\ 0 & \cos\alpha_i\sin\theta_i & \cos\alpha_i\cos\theta_i & -\sin\alpha_i \\ 0 & \sin\alpha_i\sin\theta_i & \sin\alpha_i\cos\theta_i & \cos\alpha_i \end{bmatrix} \begin{Bmatrix} 1 \\ x_g \\ y_g \\ z_g \end{Bmatrix}_{i+1} \tag{2-5}$$

In (2-1), (2-4) and (2-5) g stands for any generic point. Let's now use our results for finding ($P_{1,1}$,...., $P_{N+1,1}$) meaning (P_1,.... .., P_{N+1}) with respect to frame 1 which is attached to the ground. Consider now the coordinates of endpoint P_i with respect to frame of reference i. These coordinates are:

$$P_{i,i}=\begin{Bmatrix}1\\ a_i\\ 0\\ 0\end{Bmatrix} \tag{2-6}$$

So the problem of finding ($P_{1,1}$, $P_{2,1}$,...., $P_{N+1,1}$) while we know ($P_{1,1}$, $P_{2,2}$,..., $P_{N+1,N+1}$) is just that of several coordinate transformations applied successively to $P_{i,i}$. These transformations are indicated below

$$\begin{aligned}
&P_1 = P_{1,1} = P_{1,1}\\
&P_2 = P_{2,1} = [R_1]P_{2,2}\\
&P_3 = P_{3,1} = [R_1][R_2]P_{3,3}\\
&\cdots\cdots\cdots\cdots\cdots\cdots\\
&P_{i+1} = P_{i+1,1} = [R_1][R_2][R_3]\cdots\cdots[R_i]P_{i+1,i+1}
\end{aligned} \tag{2-7}$$

Then the positions of endpoints (P_1,..., P_{N+1}) with respect to fixed frame 1 are known because the right hand sides of equations (2-7) are formed by known parameters. Here the generic link number has been chosen as i+1 instead of i for convenience as we will see later.

Velocity Description

Now, using the same tools let's derive step 3). Assuming that a set of "velocities" ($\dot{P}_1$,...., $\dot{P}_{N+1}$) has been computed at step 2), the procedure of computing ($\dot{a}_1$,...., $\dot{\theta}_N$) has to be carried out. As can be seen from the definition itself of the "velocities", $\dot{P}_1$ will define $\dot{a}_1$, $\dot{P}_2$ will define $\dot{a}_2$, $\dot{\alpha}_1$, $\dot{\theta}_1$, $\dot{P}_3$ will define $\dot{a}_3$, $\dot{\alpha}_2$, $\dot{\theta}_2$ and so

forth. Stated in another form the problem can be proposed as follows; given a set ($\dot{a}_1,\ldots,\dot{a}_i,\dot{\alpha}_1,\ldots,\dot{\alpha}_{i-1},\dot{\theta}_1,\ldots,\dot{\theta}_{i-1}$) and $\dot{P}_{i+1}$ compute ($\dot{a}_{i+1},\dot{\alpha}_i,\dot{\theta}_i$). If we apply this procedure from the first to the last link we will have solved the initially posed problem. In order to do this, let's find the derivative of equations (2-7) with respect to the independent variable t.

$$\dot{P}_{i+1}=[\dot{R}_1][R_2]\ldots[R_i]P_{i+1,i+1}+[R_1][\dot{R}_2]\ldots[R_i]P_{i+1,i+1}+\ldots\ldots\ldots$$
$$\ldots\ldots+[R_1][R_2]\ldots[\dot{R}_i]P_{i+1,i+1}+[R_1][R_2]\ldots\ldots[R_i]\dot{P}_{i+1,i+1} \qquad (2\text{-}8)$$

Where $[\dot{R}_i]$ is,

$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ \dot{a}_i & -\dot{\theta}_i\sin\theta_i & -\dot{\theta}_i\cos\theta_i & 0 \\ 0 & -\dot{\alpha}_i\sin\alpha_i\sin\theta_i+\dot{\theta}_i\cos\alpha_i\cos\theta_i & -\dot{\alpha}_i\sin\alpha_i\cos\theta_i-\dot{\theta}_i\cos\alpha_i\sin\theta_i & -\dot{\alpha}_i\cos\alpha_i \\ 0 & \dot{\alpha}_i\cos\alpha_i\sin\theta_i+\dot{\theta}_i\sin\alpha_i\cos\theta_i & \dot{\alpha}_i\cos\alpha_i\cos\theta_i-\dot{\theta}_i\sin\alpha_i\sin\theta_i & -\dot{\alpha}_i\sin\alpha_i \end{bmatrix} \qquad (2\text{-}9)$$

and

$$\dot{P}_{i+1,i+1}=\begin{Bmatrix} 0 \\ \dot{a}_{i+1} \\ 0 \\ 0 \end{Bmatrix} \qquad (2\text{-}10)$$

With the assumptions made all terms in equations (2-8) are known except

$[\dot{R}_i]$ and $\dot{P}_{i+1,i+1}$, so we solve the equation for these two terms,

$$[\dot{R}_i]P_{i+1,i+1} + [R_i]\dot{P}_{i+1,i+1} = [R_{i-1}]^{-1}[R_{i-2}]^{-1}\cdots\cdots[R_2]^{-1}[R_1]^{-1}\{\dot{P}_{i+1} -$$

$$-[\dot{R}_1][R_2]\cdots[R_i]P_{i+1,i+1} - [R_1][\dot{R}_2]\cdots\cdots[R_i]P_{i+1,i+1} - \cdots\cdots\cdots$$

$$\cdots-[R_1][R_2]\cdots\cdots[\dot{R}_{i-1}][R_i]P_{i+1,i+1} \qquad (2\text{-}11)$$

The right hand side of equation (2-11) can be computed with the assumed data, and the result is a vector V whose components are,

$$V = \begin{Bmatrix} 0 \\ v_x \\ v_y \\ v_z \end{Bmatrix}$$

Rewriting equation (2-11),

$$[\dot{R}_i]P_{i+1,i+1} + [R_i]\dot{P}_{i+1,i+1} = V \qquad (2\text{-}12)$$

This equation splits into three equations with unknowns $\dot{a}_{i+1}$, $\dot{\alpha}_i$, $\dot{\theta}_i$

$$-a_{i+1}\dot{\theta}_i\sin\theta_i + \dot{a}_{i+1}\cos\theta_i = v_x - \dot{a}_i \qquad (2\text{-}13)$$

$$-a_{i+1}\dot{\alpha}_i\sin\alpha_i\sin\theta_i + a_{i+1}\dot{\theta}_i\cos\alpha_i\cos\theta_i + \dot{a}_{i+1}\cos\alpha_i\sin\theta_i = v_y \qquad (2\text{-}14)$$

$$a_{i+1}\dot{\alpha}_i\cos\alpha_i\sin\theta_i + a_{i+1}\dot{\theta}_i\sin\alpha_i\cos\theta_i + \dot{a}_{i+1}\sin\alpha_i\sin\theta_i = v_z \qquad (2\text{-}15)$$

Rewriting it in the matrix form,

$$\begin{bmatrix} \cos\theta_i & -a_{i+1}\sin\theta_i & 0 \\ \cos\alpha_i\sin\theta_i & a_{i+1}\cos\alpha_i\cos\theta_i & -a_{i+1}\sin\alpha_i\sin\theta_i \\ \sin\alpha_i\sin\theta_i & a_{i+1}\sin\alpha_i\cos\theta_i & a_{i+1}\cos\alpha_i\sin\theta_i \end{bmatrix} \begin{Bmatrix} \dot{a}_{i+1} \\ \dot{\theta}_i \\ \dot{\alpha}_i \end{Bmatrix} =$$

$$= \begin{Bmatrix} v_x \\ v_y \\ v_z \end{Bmatrix} - \begin{Bmatrix} \dot{a}_i \\ 0 \\ 0 \end{Bmatrix} \qquad (2\text{-}16)$$

Solving the system (2-16) by inverting the matrix we get,

$$\begin{Bmatrix} \dot{a}_{i+1} \\ \dot{\theta}_i \\ \dot{\alpha}_i \end{Bmatrix} = \begin{bmatrix} \cos\theta_i & \cos\alpha_i\sin\theta_i & \sin\alpha_i\sin\theta_i \\ \dfrac{-\sin\theta_i}{a_{i+1}} & \dfrac{\cos\alpha_i\cos\theta_i}{a_{i+1}} & \dfrac{\sin\alpha_i\cos\theta_i}{a_{i+1}} \\ 0 & \dfrac{-\sin\alpha_i}{a_{i+1}\sin\theta_i} & \dfrac{\cos\alpha_i}{a_{i+1}\sin\theta_i} \end{bmatrix} \begin{Bmatrix} v_x - \dot{a}_i \\ v_y \\ v_z \end{Bmatrix} \qquad (2\text{-}17)$$

The matrix in (2-17) will be denoted $[S_i]$ and U_i will denote the vector:

$$U_i = \begin{Bmatrix} \dot{a}_{i+1} \\ \dot{\theta}_i \\ \dot{\alpha}_i \end{Bmatrix}$$

Thus we have found the way in which given ($\dot{a}_1, \ldots, \dot{\theta}_{i-1}$) and $\dot{P}_{i+1}$ we compute U_i. So applying this procedure successively as indicated in (2-18) we can find $U_1, U_2, \ldots\ldots, U_N$

$$\dot{a}_1 = [0\ 1\ 0\ 0]\ \dot{P}_1$$

$$U_1 = [S_1]\{\dot{P}_2 - \dot{P}_{1,1}\}$$

$$U_2 = [S_2]\left\{[R_1]^{-1}\left\{\dot{P}_3 - [\dot{R}_1][R_2]P_{3,3}\right\} - \dot{P}_{2,2}\right\} \qquad (2\text{-}18)$$

$$U_3 = [S_3]\left\{[R_2]^{-1}[R_1]^{-1}\left\{\dot{P}_4 - [\dot{R}_1][R_2][R_3]P_{4,4} - [R_1][\dot{R}_2][R_3]P_{4,4}\right\} - \dot{P}_{3,3}\right\}$$

And we continue to generate the angular(extensional) "velocities" of the joints for as many links as we need.

An additional feature which allows us to widen the group of manipulators which can use this algorithm is the following: We have previously assumed that the end point P_0 of link 1 was clamped to the ground, and that this link was parallel to the x axis. This new feature which can be seen in Figure 5 presents two additional degrees of freedom which are translations along the y and z axes of the previously clamped point P_0. The analytical derivations of steps 1) and 3) hold except that we should add the a_y and a_z translation components to all y and z components of the positions ($P_1, \ldots, P_{N+1}$), and, further, we should substract $\dot{a}_y$ and $\dot{a}_z$ from all components of the "velocities" ($\dot{P}_1, \ldots, \dot{P}_{N+1}$) respectively. Steps 1) and 3) are thus concluded.

Guiding the Manipulator

The remaining step 2) in this algorithm makes use of the positions ($P_1, \ldots, P_{N+1}$) to find a set of "velocities" or influences ($\dot{P}_1, \ldots, \dot{P}_{N+1}$). The way in which this step was solved makes use of the following assumptions. The end point P_{N+1} of the manipulator moves in a straight line towards its destination point P_{dest} whenever there is not an obstacle close to P_{N+1}. When there is an obstacle between P_{N+1} and P_{dest}, and P_{N+1} is close to an obstacle, the motion of P_{N+1} will be

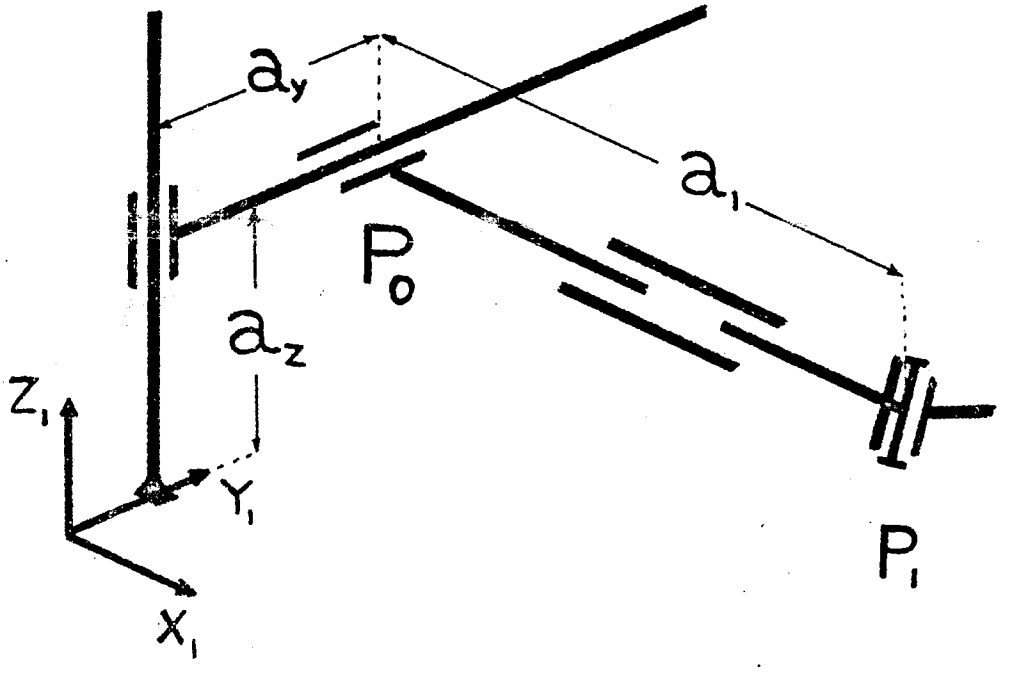

Figure 5. Additional Degrees of Freedom

the sum of an influence of approach to P_{dest} and an influence of rejection by the obstacle, which will result in a total influence $\dot{P}_{N+1}$ which is fairly parallel to the surface of the obstacle. In a later paragraph we will describe the procedure accurately. The motion of P_N is determined by the condition that the link N may be stretched, contracted, or fixed, and by the proximity of obstacles. Note that the destination point P_{dest} does not influence P_N directly, but as P_N is influenced by P_{N+1}, then P_N is indirectly influenced by P_{dest} as are all P_i's. So $\dot{P}_i$ is influenced by $\dot{P}_{i+1}$, and by the close obstacles. Repeating this step for as many times as necessary the set $(\dot{P}_1,.., \dot{P}_{N+1})$ will be generated. Before studying how we generate $(\dot{P}_1,..., \dot{P}_{N+1})$ in further detail, we first will consider a general obstacle and see how we construct the rejection influence of the manipulator links and joints by such an obstacle. A rejection influence from an obstacle will be a vector g applied at one or two consecutive joints which when added to the influences of other obstacles and to the components necessary for ensuring physical continuity, will determine the total influence $\dot{P}_i$ of that joint. The criteria for determining the magnitude and direction of vector g is as follows: Suppose we have a link i with endpoints P_{i-1} and P_i. Three possibilities can occur; a) point P_i is the closest to the obstacle; b) segment $P_{i-1}P_i$ is the closest to the obstacle; c) point P_{i-1} is the closest to the obstacle. In possibility a) shown in Figure 6 the vector g is constructed such that: it passes by P_i, it is perpendicular to the surface of the obstacle at the surface's nearest point and its magnitude is a continuous and decreasing function of the minimun distance from the obstacle to P_i. In order to ensure that there is no possibility of contact with the obstacle,

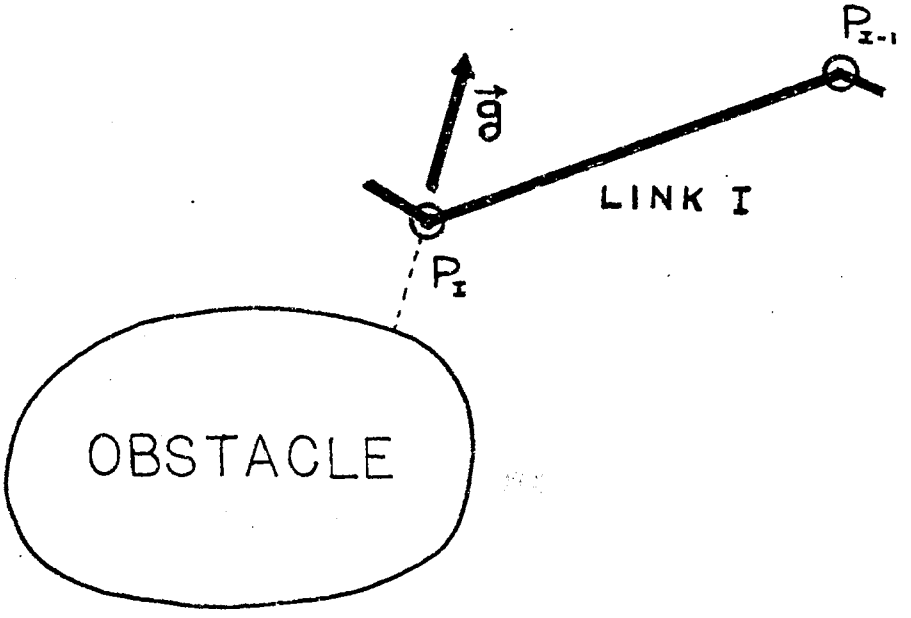

Figure 6. Influence of an Obstacle on P_i

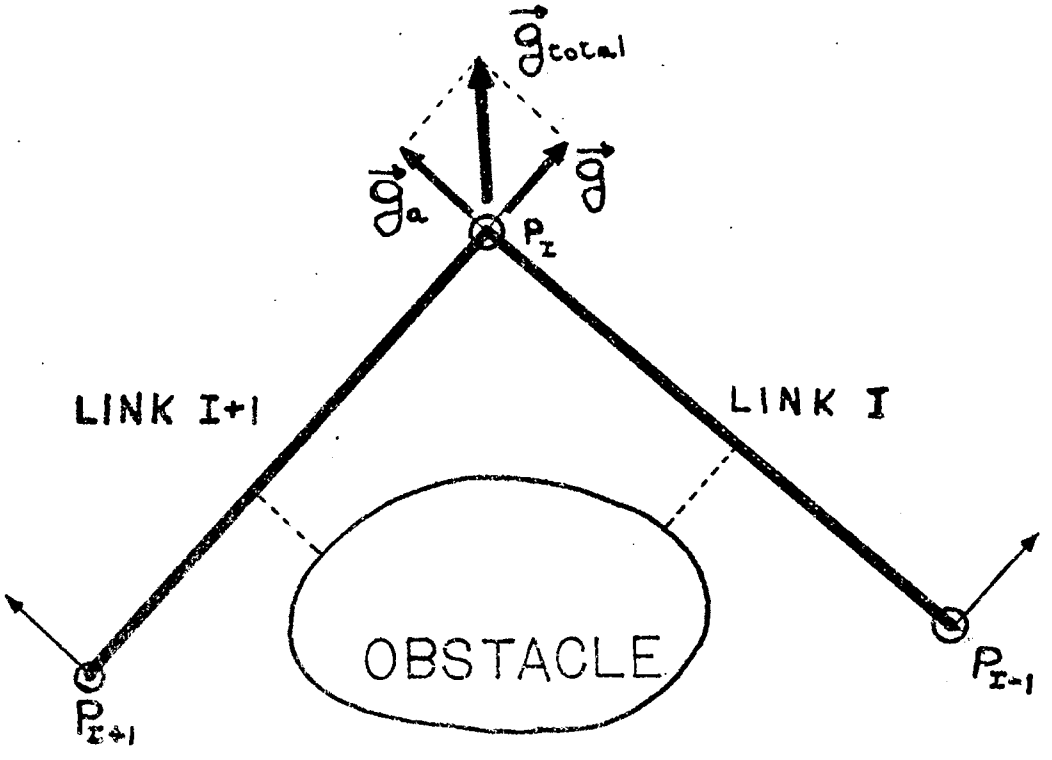

Figure 7. Influence of an Obstacle on $P_{i-1}P_i$

we choose this function to be infinitely large when the minimun distance from the obstacle to P_i approaches to zero, and in order to avoid influence by the obstacle when the link i is far, we choose this function to be zero for a certain distance or greater. In possibility b) the common normal between the segment $P_{i-1}P_i$ and the surface of the obstacle will determine the direction of vector g, its magnitude will be determined as in possibility a). The vector g will be applied to P_i and to P_{i-1} in the following way: if this obstacle influenced segment $P_{i+1}P_i$ of link i+1 as well, then the influence on P_i will be half the vectorial sum of the influences in the i+1 and i links, as shown in Figure 7. Here g_a stands for the g computed in the previous step at link i+1. If this obstacle influenced link i+1 at one of its endpoints (if possibility a) or c) applied to link i+1), then no modification of the g computed previously will be made as shown in Figure 8. In possibility c) we take no action because we will consider P_{i-1} as part of the i-1 link.

So each obstacle k will generate an influence on P_i which we designate by g_{ik}. Let us return now to our goal on how we generate ($\dot{P}_1$, ..., $\dot{P}_{N+1}$). Introducing coordinates as P_{dest} (x_d, y_d, z_d) and P_i (x_i, y_i, z_i), the influence towards P_{dest} on point P_{N+1} will be a vector w whose detail is shown below,

$$w = \begin{Bmatrix} b(x_d - x_{N+1}) \\ b(y_d - y_{N+1}) \\ b(z_d - z_{N+1}) \end{Bmatrix} \qquad (2\text{-}19)$$

where b is an arbitrary constant. Then finally,

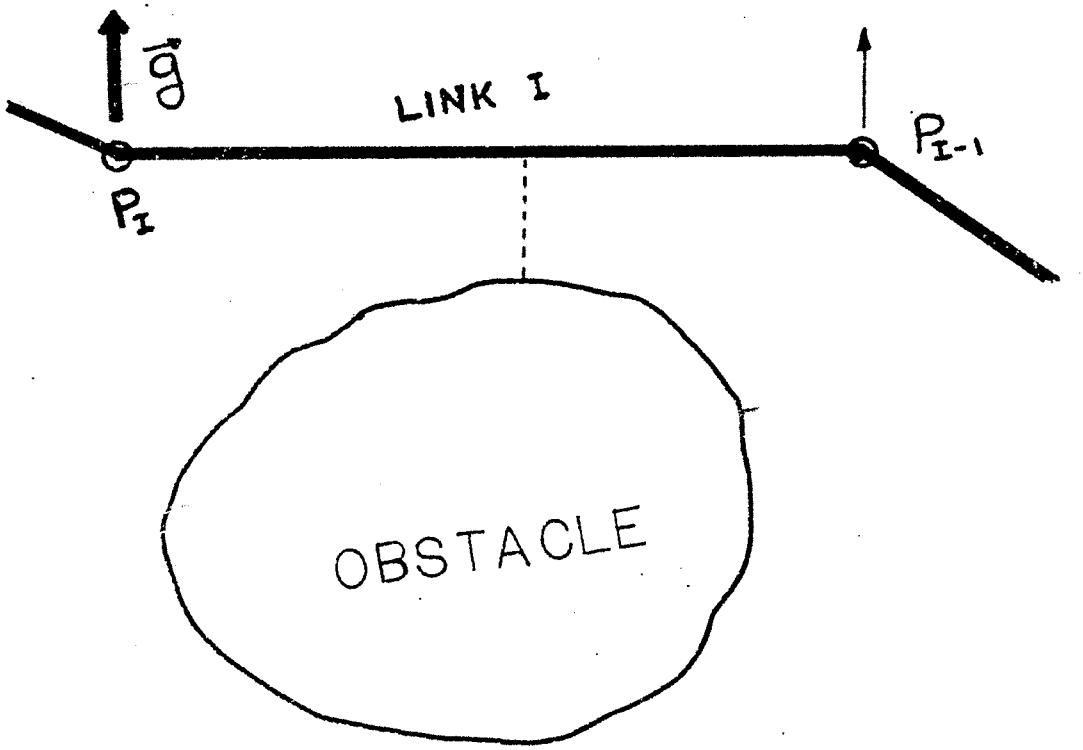

Figure 8. Influence of an Obstacle on $P_{i-1}P_i$

$$\dot{P}_{N+1} = w + \sum_{\kappa} g_{N+1,k} \tag{2-20}$$

where the sumation is applied to all the obstacles. Suppose now that we have $\dot{P}_{i+1}$ and we want to find $\dot{P}_i$. Let us define

$$u_{i+1,i} = \frac{P_{i+1} - P_i}{\left|P_{i-1} - P_i\right|} \tag{2-21}$$

The projection of vector $\dot{P}_{i+1}$ on vector $u_{i+1,i}$ is a vector whose length is L_{i+1} and is computed as

$$L_{i+1} = \dot{P}_{i+1} \cdot u_{i+1,i} \tag{2-22}$$

Let

$$V_{obs,i} = \sum_{\kappa} g_{i,k} \tag{2-23}$$

The vector $V_{obs,i}$ cannot alone be $\dot{P}_i$ because P_{i+1} and P_i are parts of a deformable link and have a constraint. Let $\dot{a}_{i+1}$ be the stretching rate of link i+1. Letting

$$L_{obs,i} = V_{obs,i} \cdot u_{i+1,i} \tag{2-24}$$

as most of the time

$$L_{i+1} - L_{obs,i} \neq \dot{a}_{i+1} \tag{2-25}$$

we define a variable c such that physical compatibility is preserved

$$L_{i+1} = L_{obs,i} + \dot{a}_{i+1} + c \tag{2-26}$$

Then

$$c = L_{i+1} - L_{obs,i} - \dot{a}_{i+1} \tag{2-27}$$

Finally

$$\dot{P}_i = \sum_{\kappa} g_{i,k} + c\, u_{i+1,i} \tag{2-28}$$

By applying the same procedure from P_N to P_1 we find all ($\dot{P}_1$,..... .., $\dot{P}_{N+1}$). An observation remains to be expressed with respect to $\dot{a}_{i+1}$. Suppose that link i+1 is an extensible link and let $a_{i+1,max}$ and $a_{i+1,min}$ be the maximun and minimun lengths of link i+1 respectively. In order to find c we have to make assumptions which are realistic. These are: whenever possible

$$\dot{a}_{i+1} = L_{i+1} - L_{obs,i} \tag{2-29}$$

therefore $c = 0$ and

$$\dot{P}_i = \sum_{\kappa} g_{i,k} \tag{2-30}$$

By possible we mean that $\dot{a}_{i+1} > 0$ and $a_{i+1} \geq a_{i+1,max}$ do not hold both simultaneously and $\dot{a}_{i+1} < 0$ and $a_{i+1} \leq a_{i+1,min}$ do not hold simultaneously as well. The meaning is clear. If the link is fully extended $\dot{a}_{i+1}$ can only be zero or negative. Similarly if the link is fully contracted $\dot{a}_{i+1}$ can only be zero or positive. If any of the impossible conditions hold, then $\dot{a}_{i+1}$ will be taken as zero and

$$c = L_{i+1} - L_{obs,i} \tag{2-31}$$

Although we have given the general method for finding the successive influences ($\dot{P}_1, \ldots, \dot{P}_{N+1}$), there is a special combination of parameters in which $\dot{P}_2$ will have to be different from the $\dot{P}_2$ given in the previous procedure because otherwise the assumed influences will not be physically realizable. We should choose a vector $\dot{P}_2$ which will be tangent to a sphere of center (a_1, a_y, a_z) and radius a_2 whenever a_1, a_2, a_3, a_y and a_z are fixed at the same time, and whose projection on $u_{3,2}$ will be the same as the projection of $\dot{P}_3$. Let ($\dot{x}_2$, $\dot{y}_2$, $\dot{z}_2$) be the components of the originally computed $\dot{P}_2$ and let the unit normal to the sphere of radius a and whose polar coordinates are θ_1 and α_1 be,

$$n = \begin{Bmatrix} \cos\theta_1 \\ \sin\theta_1\cos\alpha_1 \\ \sin\theta_1\sin\alpha_1 \end{Bmatrix} \qquad (2\text{-}32)$$

In order to find the projection of $\dot{P}_2$ on the tangent plane at the surface of the sphere, we first find the projection of $\dot{P}_2$ on the normal n and then substract this projection from $\dot{P}_2$. The projection on n is,

$$p = (\dot{P}_2 \cdot n) n \qquad (2\text{-}33)$$

Then the projection on the tangent plane $\dot{P}_{2pr}$ will be

$$\dot{P}_{2pr} = \dot{P}_2 - p = \dot{P}_2 - (\dot{P}_2 \cdot n) n = \left[[I] - [n \otimes n] \right] \dot{P}_2 \qquad (2\text{-}34)$$

where the symbol $\otimes$ indicates tensor or dyadic product. In coordinates (2-34) gives

$$\begin{Bmatrix} \dot{x}_{2pr} \\ \dot{y}_{2pr} \\ \dot{z}_{2pr} \end{Bmatrix} = \begin{bmatrix} \sin^2\theta_1 & -\cos\theta_1\sin\theta_1\cos\alpha_1 & -\cos\theta_1\sin\theta_1\sin\alpha_1 \\ -\cos\theta_1\sin\theta_1\cos\alpha_1 & (1-\sin^2\theta_1\cos^2\alpha_1) & -\sin^2\theta_1\cos_1\sin\alpha_1 \\ -\cos\theta_1\sin\theta_1\sin\alpha_1 & -\sin^2\theta_1\cos_1\sin\alpha_1 & (1-\sin^2\theta_1\sin^2\alpha_1) \end{bmatrix} \begin{Bmatrix} \dot{x}_2 \\ \dot{y}_2 \\ \dot{z}_2 \end{Bmatrix} \tag{2-35}$$

So far we have found the direction of the modified $\dot{P}_2$. In order to find its magnitude we proceed as follows,

$$b = \frac{\dot{P}_3 \cdot u_{3,2}}{\dot{P}_{2pr} \cdot u_{3,2}} \tag{2-36}$$

and finally

$$P_{2modified} = b\,\dot{P}_{2pr} \tag{2-37}$$

In the Appendix we will develop the details of the computations for finding $g_{i,k}$ for obstacles of different shapes. The problem of finding the common normal between a segment and a surface, or the normal from a point to a surface is simply a problem of Analytic Geometry. The same happens with computing the distance from a point or a line to a surface. In the Appendix we compute vector $g_{i,k}$ for the following elemental obstacles: 1) triangular plane element; 2) sphere; 3) cylinder. The reason we choose these elements is that any obstacle can be approximated by a finite number of these elements. In the resulting polyhedron formed by triangular faces, the edges are limited cylinders of radius equal to zero, and the vertices are spheres also of radius equal to zero. So using these three elements the vector g can be computed for a great variety of obstacles. Thus, step 2) is concluded.

CHAPTER III

RESULTS AND ANALYSIS

In this chapter, we will outline a method in which the algorithm we have developed in the previous chapter is applied to practical problems, and we will illustrate an example of it.

The correct procedure for testing an algorithm such as the one we have developed is to apply it directly to a computer controlled manipulator. Such a manipulator should have in each mobile revolute pair or cylinder pair devices for transforming the angles and/or the extensions of the joints into some form amenable to data processing. With our current technology, electronic digital signals are the most commonly used. Such devices are called shaft encoders and are commercially available in a wide variety of types. In order to avoid the computation of sines and cosines of several angles, shaft encoders could be built which could encode the shaft rotation directly into a digital signal which is the sine and cosine of the rotated angle. So far we have indicated the way in which information is obtained from the manipulator as input to our algorithm. To use the output of our algorithm we must transform these electronic digital signals into some form which will be able to exert a motion on the joints of our manipulator. The most conventional ways of doing this are to convert the electronic digital signal into an electronic analog signal using D/A converters, then to amplify it until it can drive an electric motor at each joint or

convert it to a hydraulic signal with an electrohydraulic servovalve and drive hydraulic motors at the joints.

The way in which such a system would operate during each cycle would be as follows: the computer "reads" from the shaft encoders, computes, and "writes" in the actuators or motors. By "writing" we mean that the computed information flows outward from the computer to the actuators, commanding them to move al the computed speed until the computer writes again. The interval of real time between the start of each cycle is dependent on the computer speed and on the manipulator dynamics. In control systems vocabulary a system like the previous one would be referred to as a "sampled data control system".

One question remains. What is the relation between the computed ($\dot{a}_1,\ldots, \dot{\theta}_N$) and the real angular and extensional velocities ($\dot{a}_{1real}$,....., $\dot{\theta}_{Nreal}$) at which the joints should move? If we multiply the set ($\dot{a}_1,\ldots, \dot{\theta}_N$) by a constant the manipulator will move by the same trajectory but at a different speed. So if we neglect the manipulator dynamics and assume that the speed is only limited by the actuator characteristics, the fastest way to move the manipulator would be when one of the actuators is close to saturation, this meaning at maximun speed. In order to find the set ($\dot{a}_{1real},\ldots., \dot{\theta}_{Nreal}$) that will move at the maximun attainable speed, we compute a number e as follows,

$$e = \text{Max} \left(\frac{\dot{a}_1}{\dot{a}_{1max}} , \ldots\ldots\ldots\ldots, \frac{\dot{\theta}_N}{\dot{\theta}_{Nmax}} \right) \qquad (3\text{-}1)$$

So e is the largest of all numbers enclosed by the parenthesis in (3-1). To find the real speed we should divide the computed speed by the number e. So,

$$\dot{a}_{1real} = \frac{\dot{a}_1}{e}, \ldots\ldots\ldots\ldots, \quad \dot{\theta}_{Nreal} = \frac{\dot{\theta}_N}{e} \qquad (3\text{-}2)$$

As our limited resources preclude the access to a computer controlled manipulator, in order to test the algorithm, we simulate the manipulator, its actuators and its encoders with the digital computer itself. To simulate these three elements is very simple because their net effect from the output of the computer to the input is that the inputs ($a_1, \ldots, \theta_N$) are the integrals of the outputs ($\dot{a}_1, \ldots, \dot{\theta}_N$) with respect to the independent variable t. As the simulation is not made in real time and as we are interested only in the trajectory generation we assume the number e equal to one throughout the simulation. Out of the several simulations run, one of them will be illustrated which is planar, thus having the advantage of easy interpretation.

The chosen manipulator has six links with fixed link lengths. These are $a_1 = 0$, $a_2 = a_3 = \ldots\ldots = a_7 = 2'$ and $\alpha_1 = \ldots\ldots = \alpha_6 = 0$, $a_y = 0$, $a_z = 0$. The obstacles arbitrarily chosen were three cylinders with vertical axes (seen as circles in the drawings) with the following centers (4.4, 5.), (9., 5.5) and (7.5, 1.5) all of which had radii equal to 1.5. The destination point P_{dest} of the endpoint P_7 was arbitralily chosen behind the cylinders from the initial position but with the condition that P_{dest} is close enough that it is within the grasp of the manipulator. This point was (6.8, 8.). The initial θ's were $\theta_1 = b$, $\theta_2 = -2b$, $\theta_3 = 2b$, $\theta_4 = -2b$, $\theta_5 = 2b$, $\theta_6 = -2b$ with $b = 1.44$ radians close to pi/2 radians for making the manipulator folded. The integration was carried out using the Euler method because it proceeds in the same way as a sampled data control does (the

derivatives of the parameters of the manipulator with respect to t remain constant during the cycle). Figures 9 to 13 show the trajectories of points P_3, P_4, P_5, P_6 and P_7. P_1 and P_2 are not shown because they will be a point and an arc of circle respectively. Figures 14 to 34 show the successive positions of the manipulator from its initial position until it reaches its destination point.

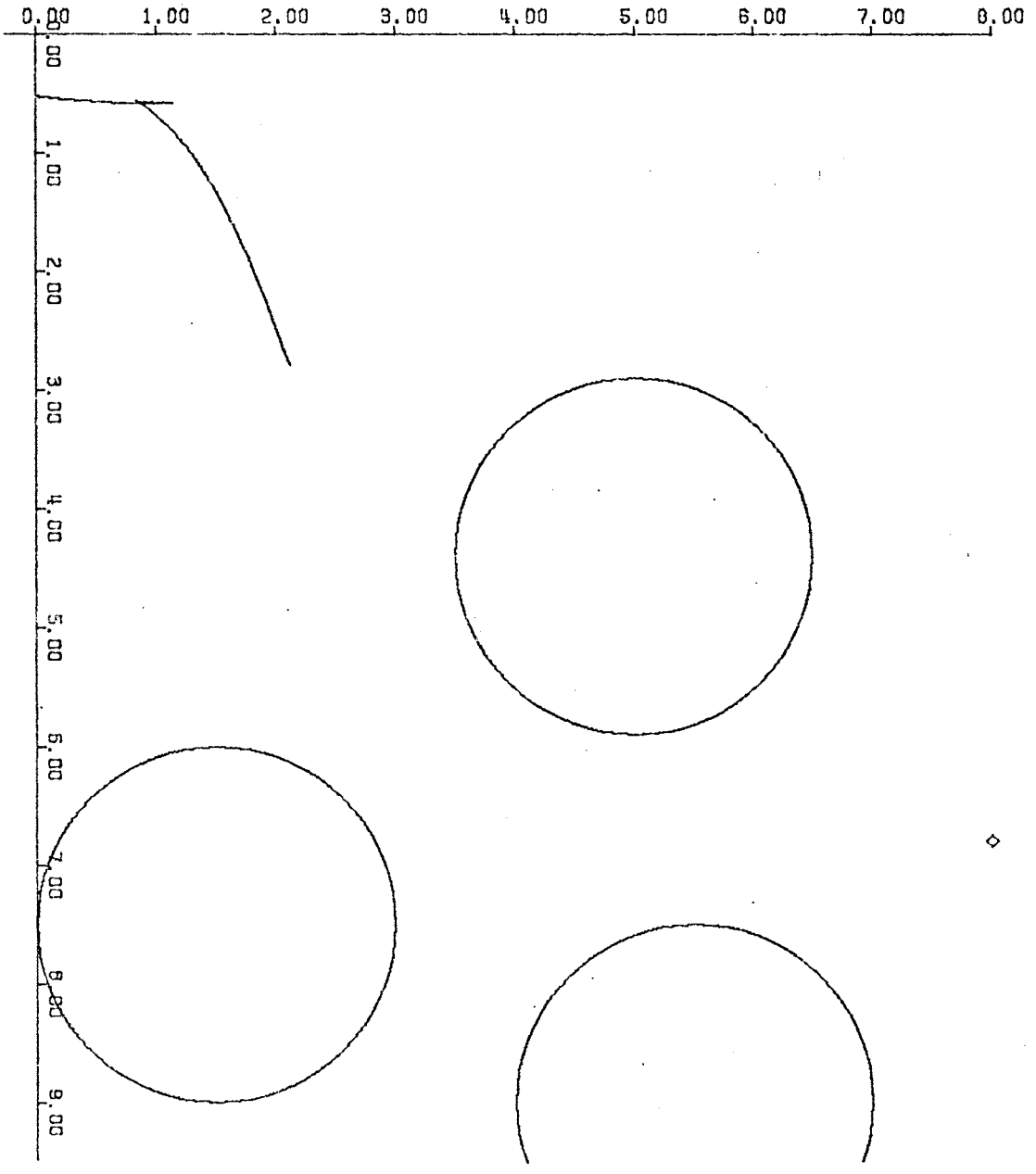

Figure 9. Trajectory

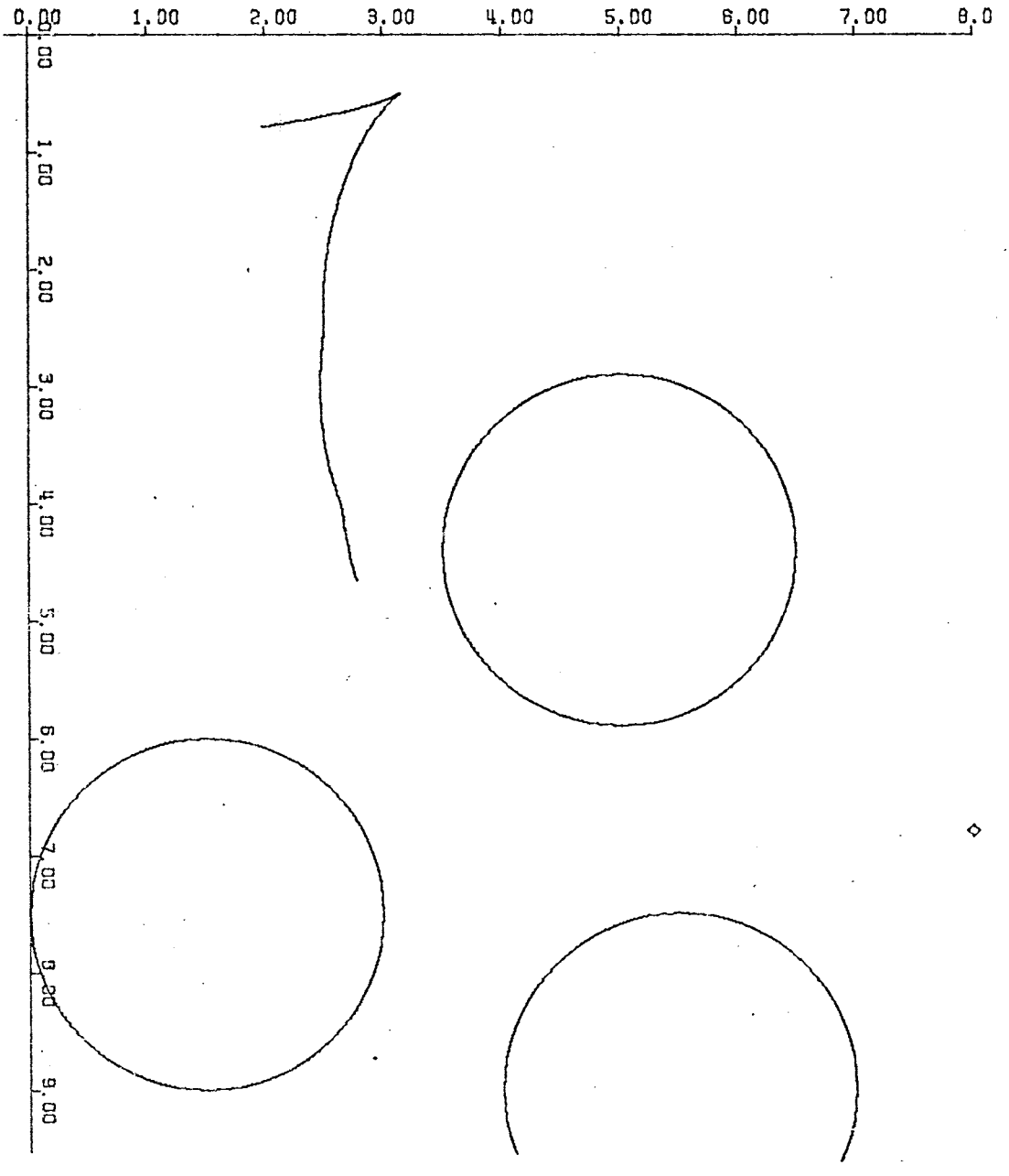

Figure 10. Trajectory

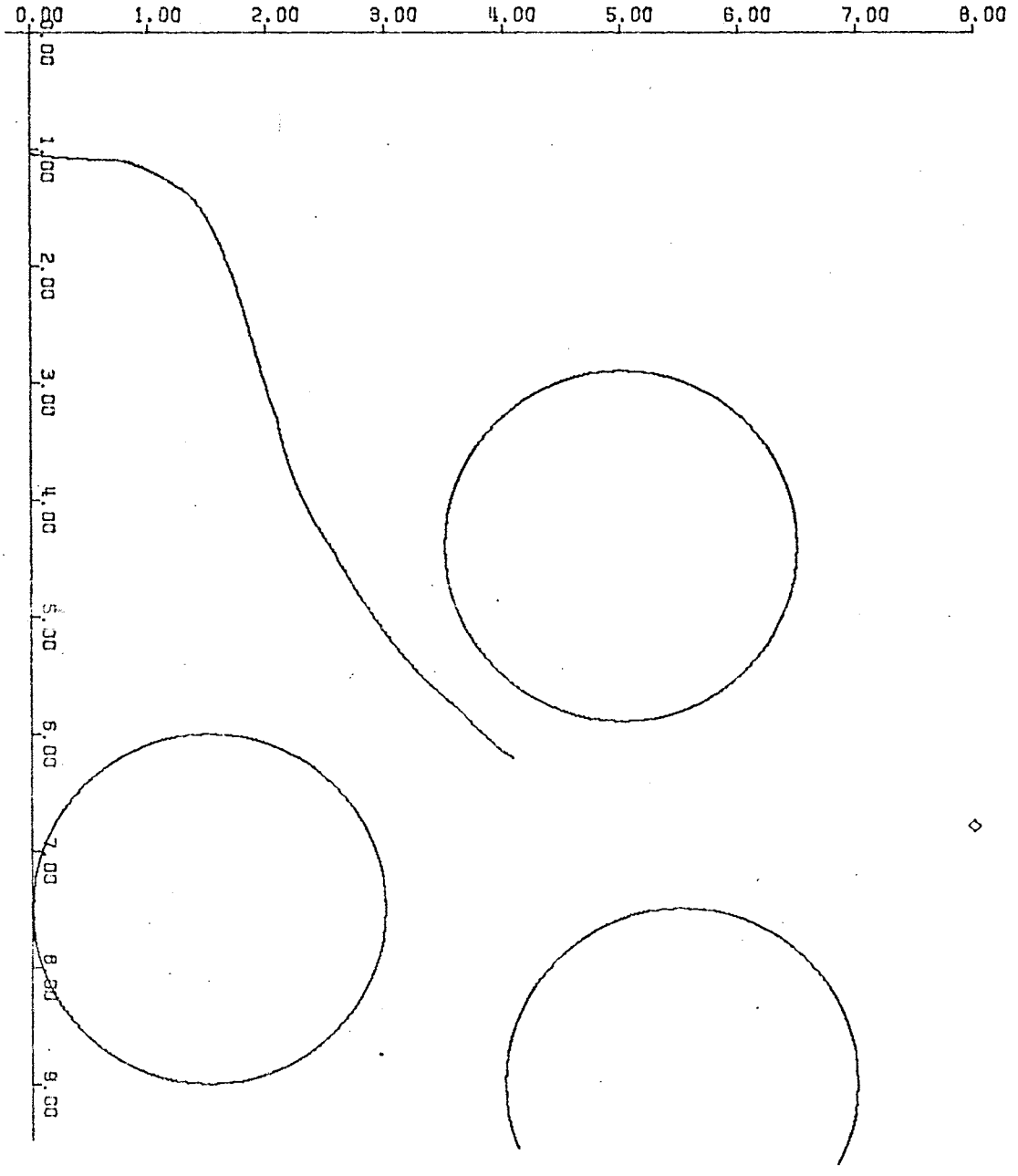

Figure 11. Trajectory

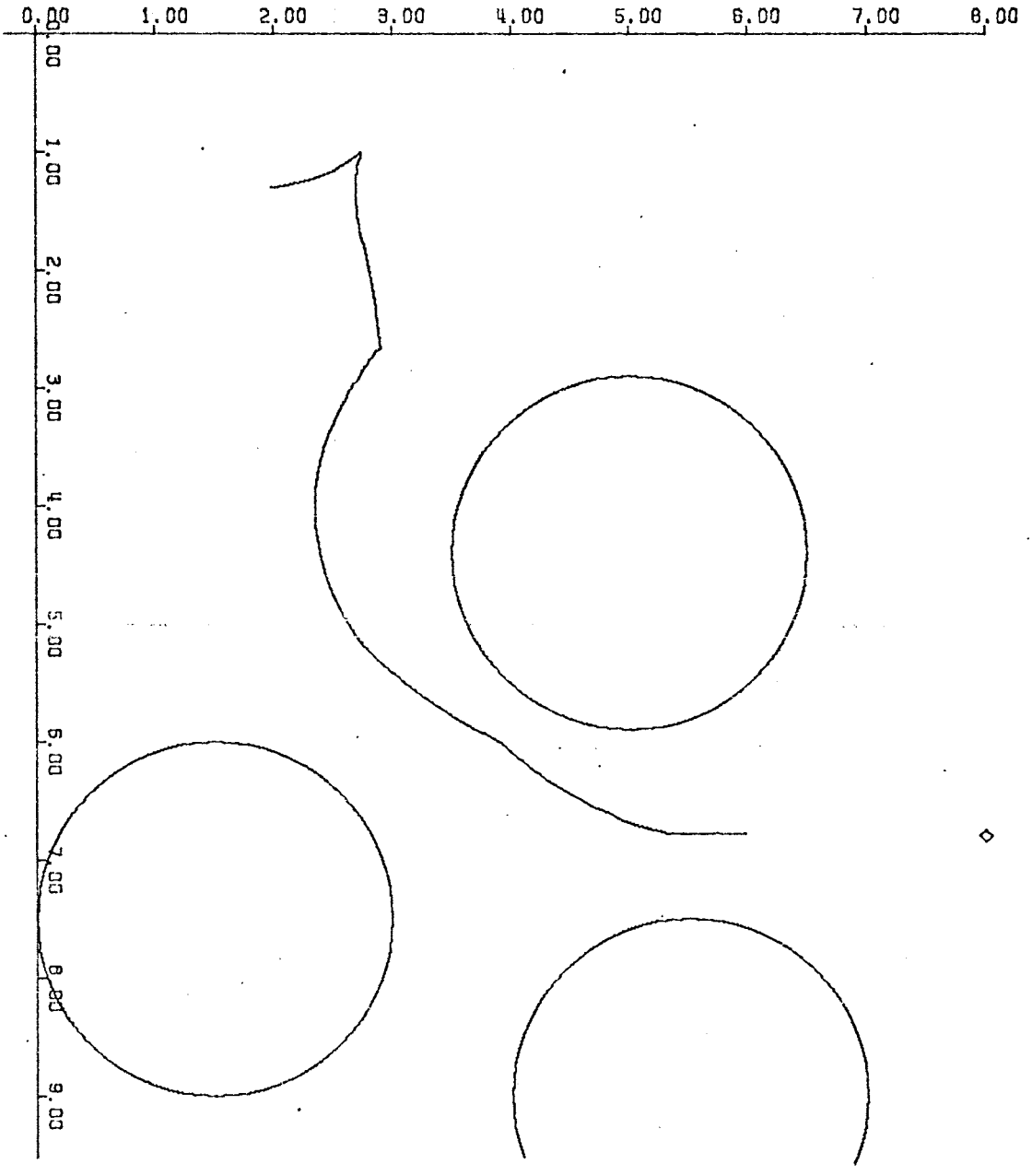

Figure 12. Trajectory

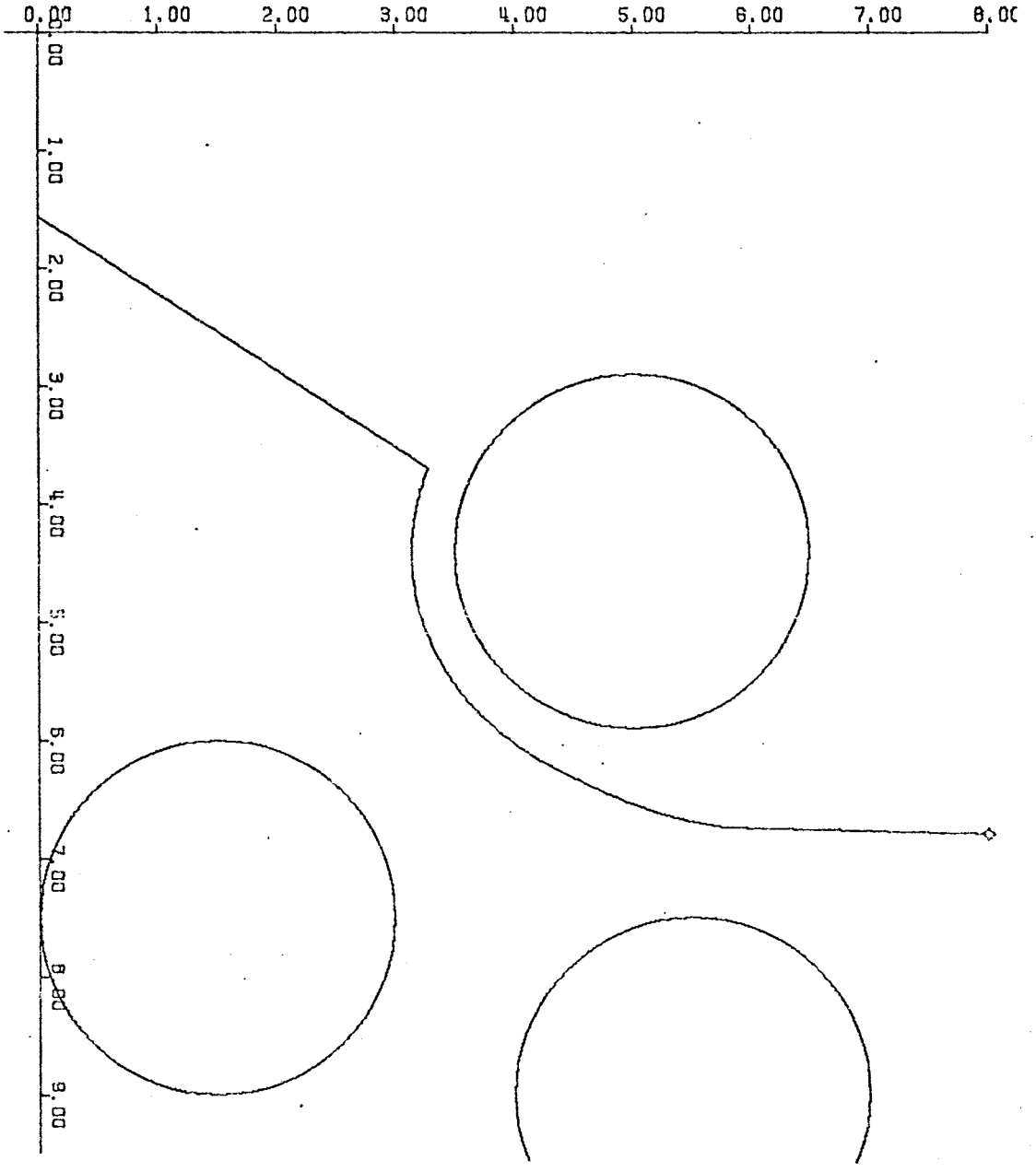

Figure 13. Trajectory

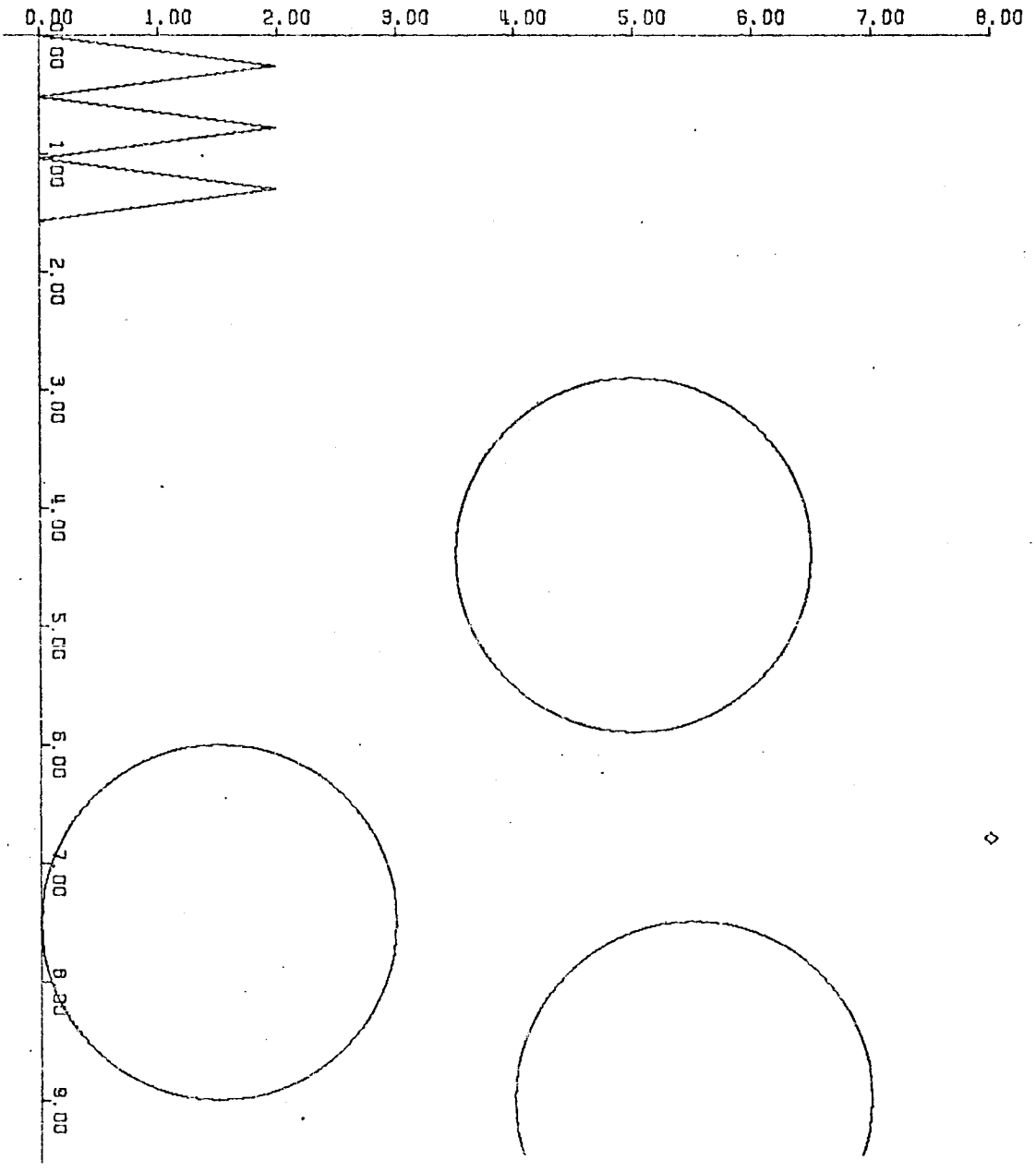

Figure 14. Succesive Positions

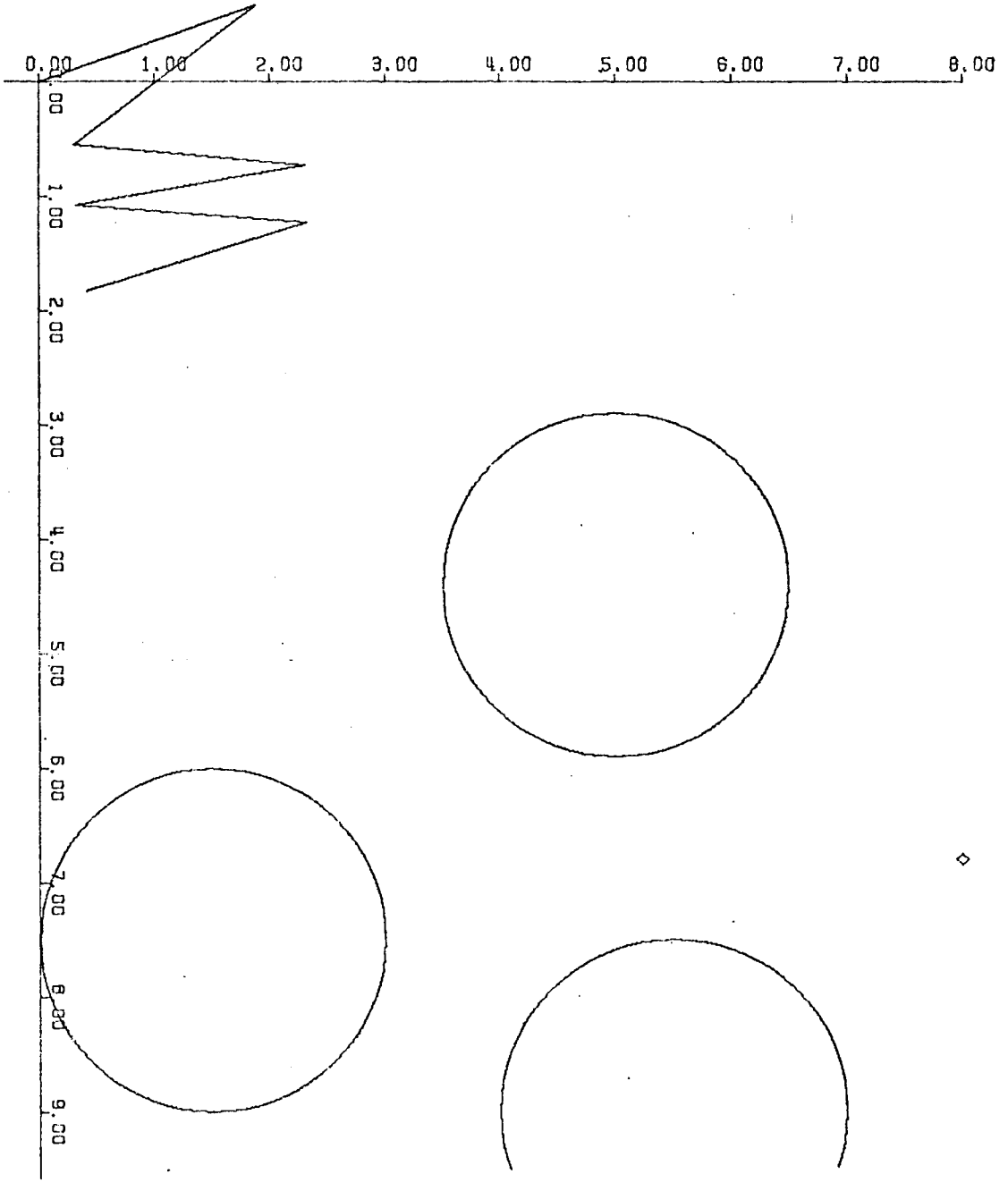

Figure 15. Successive Positions

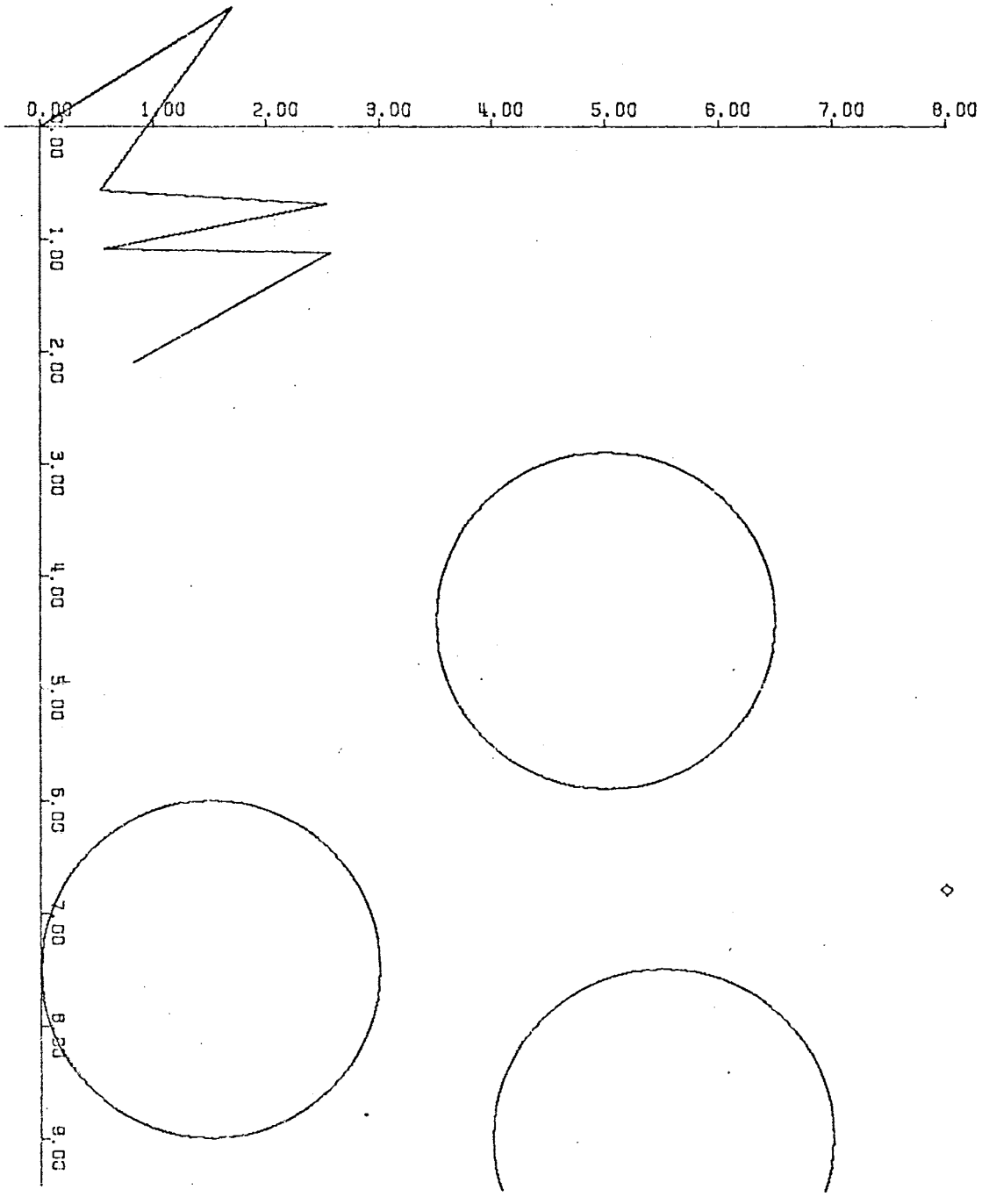

Figure 16. Successive Positions

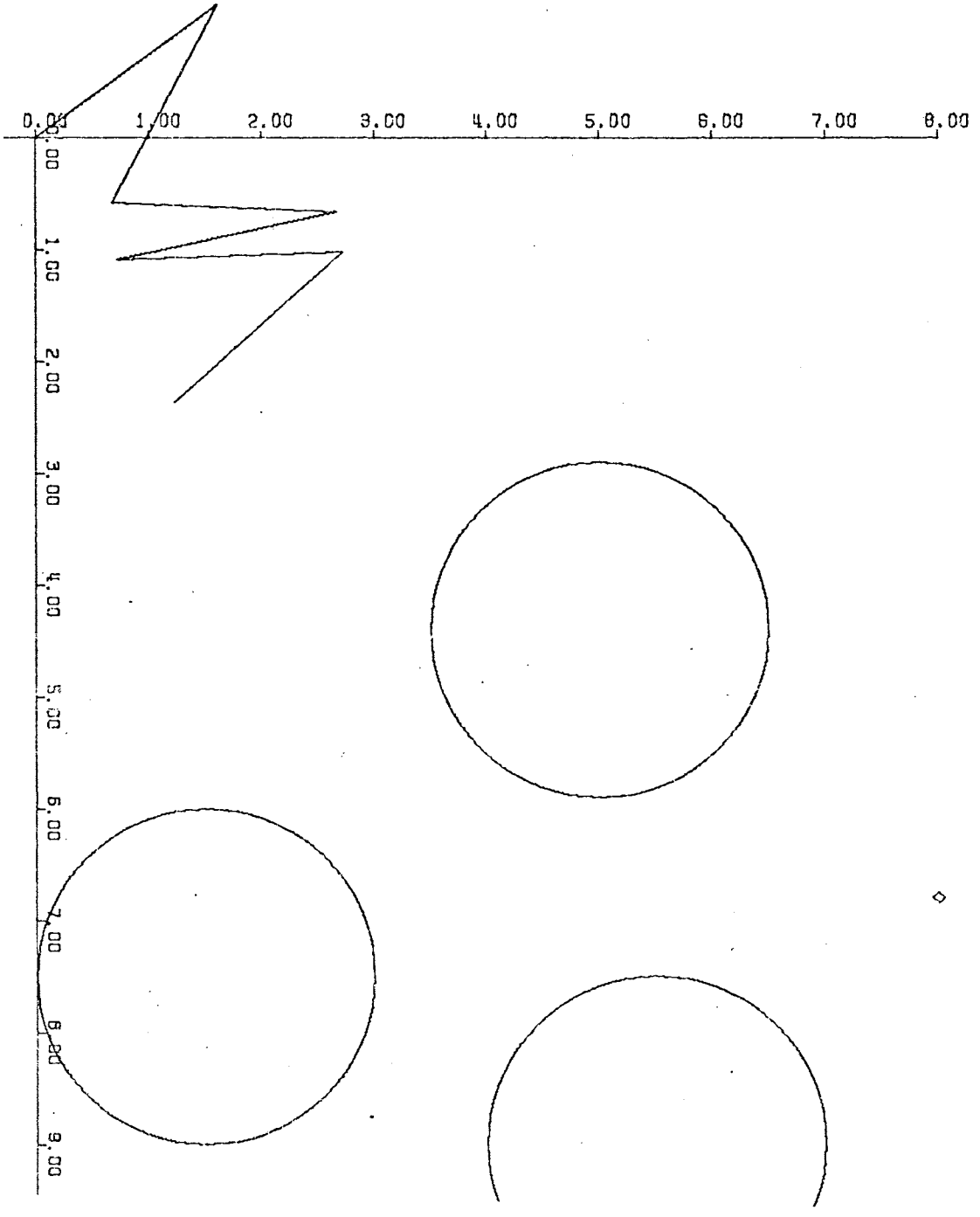

Figure 17. Successive Positions

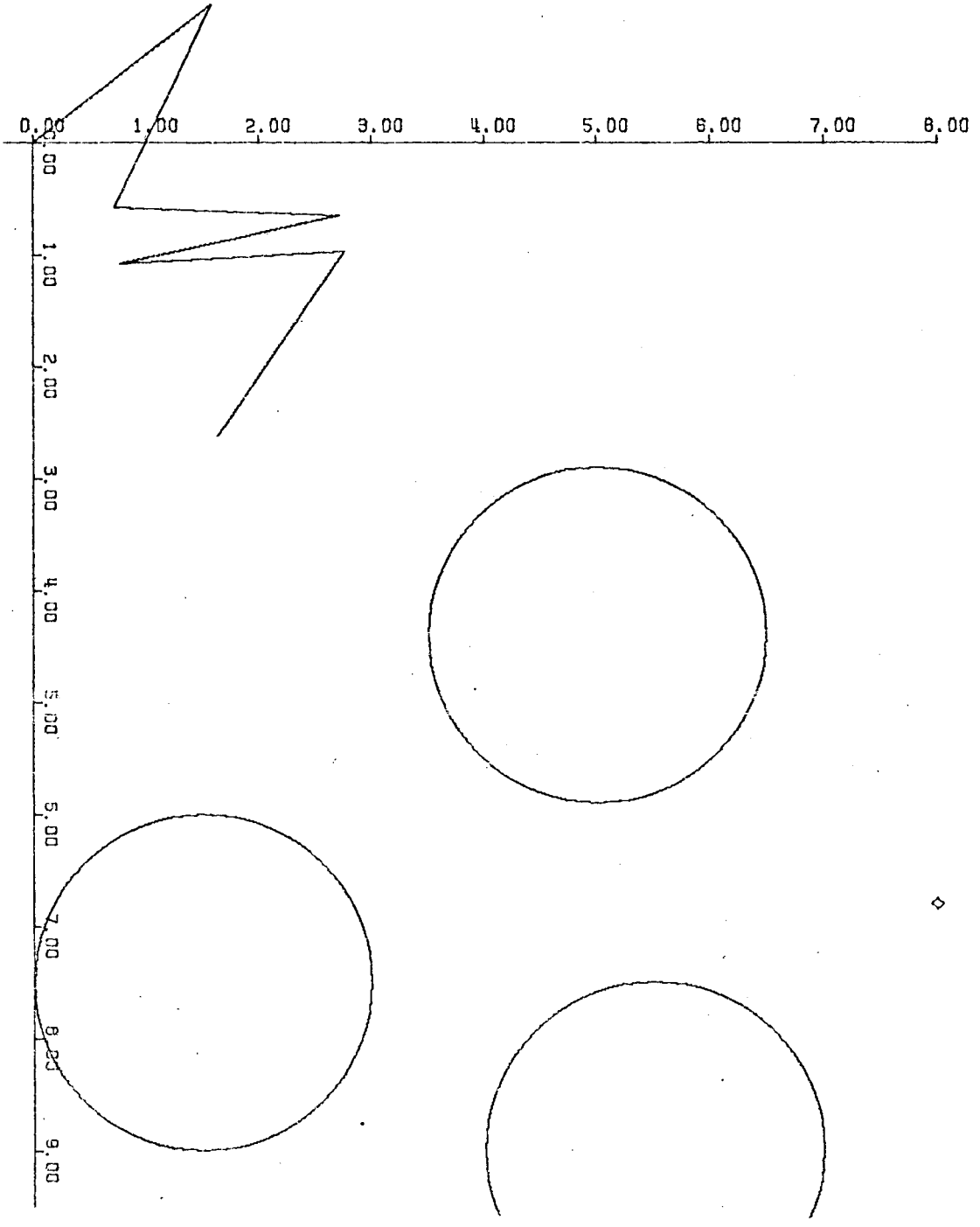

Figure 18. Successive Positions

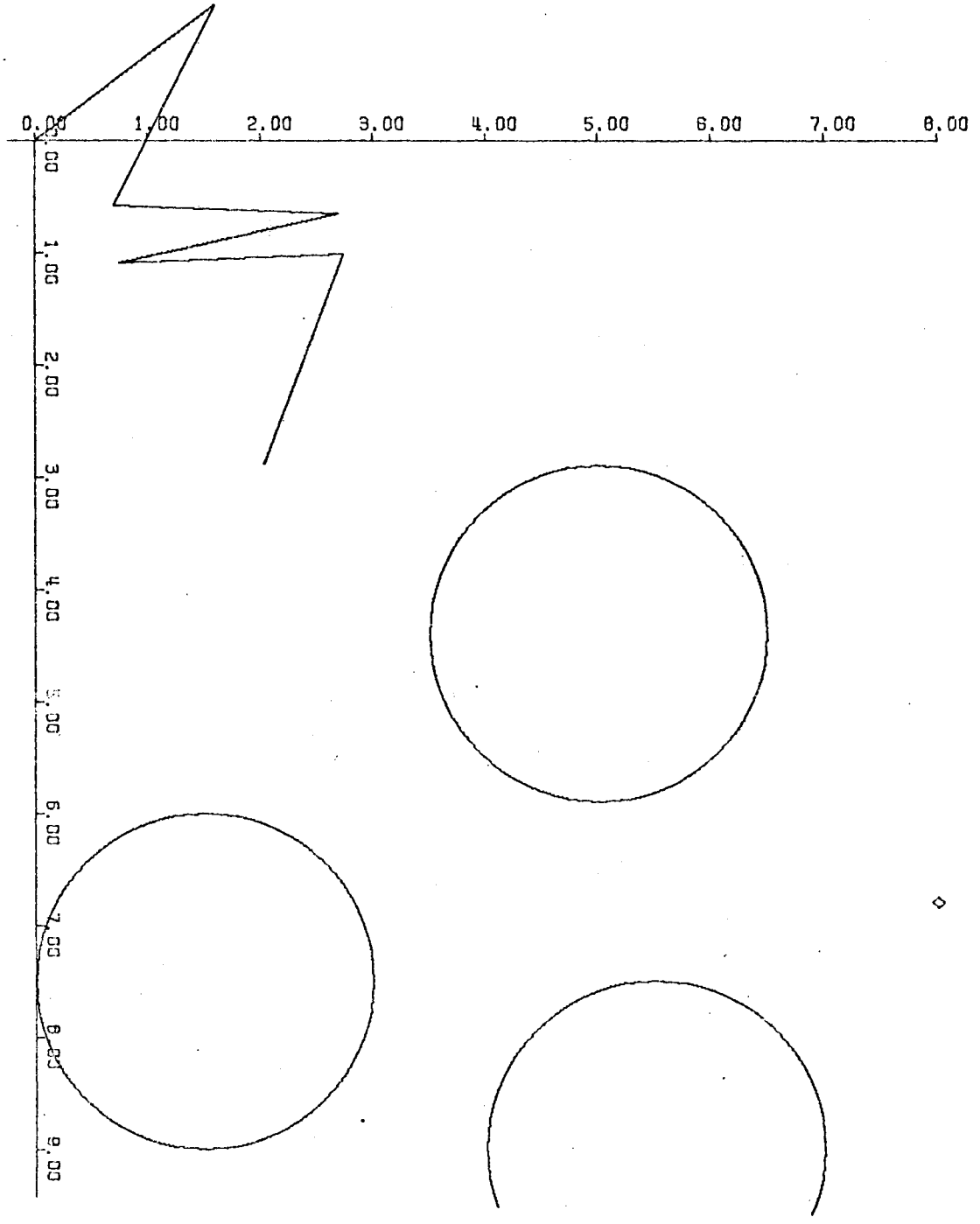

Figure 19. Successive Positions

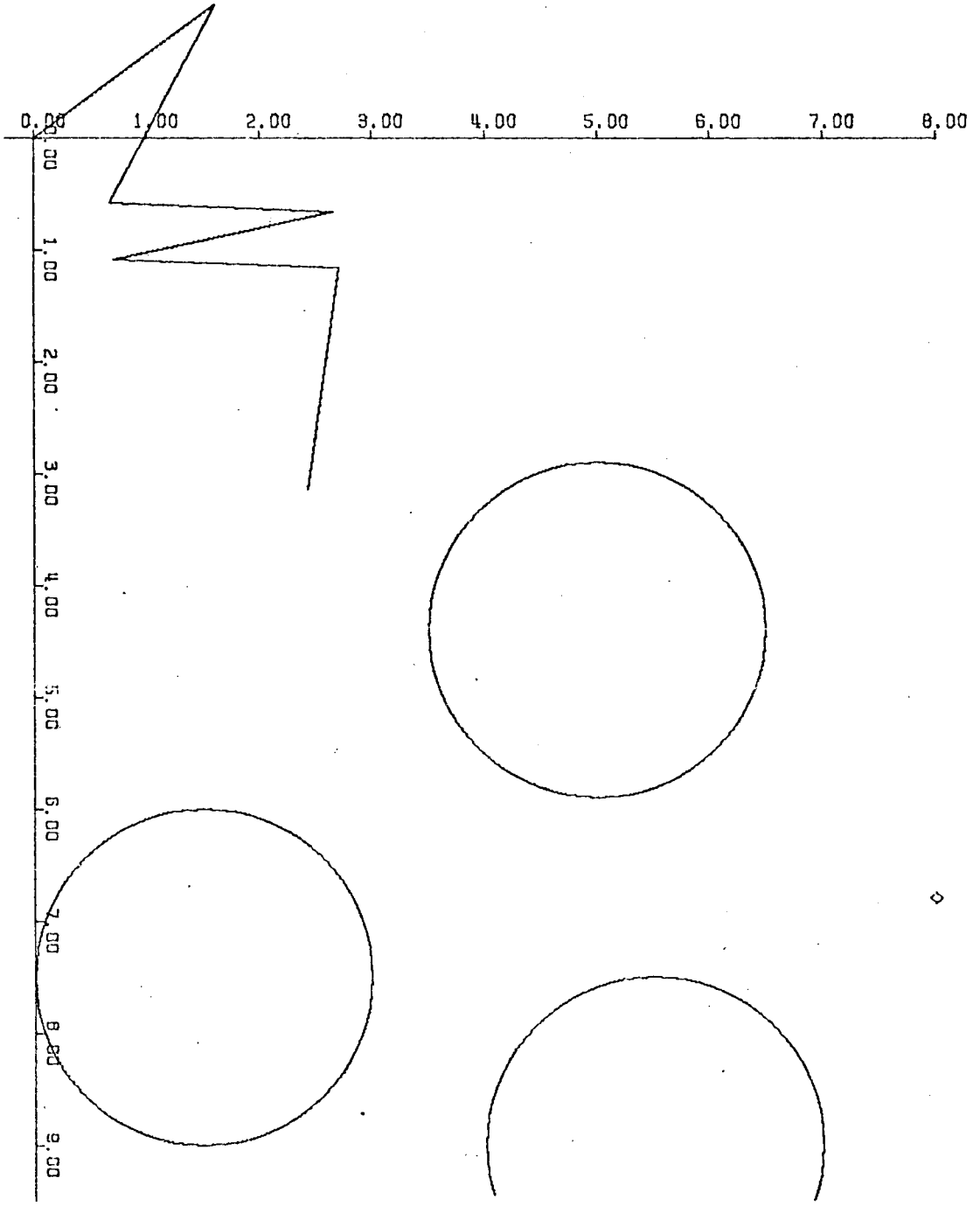

Figure 20. Successive Positions

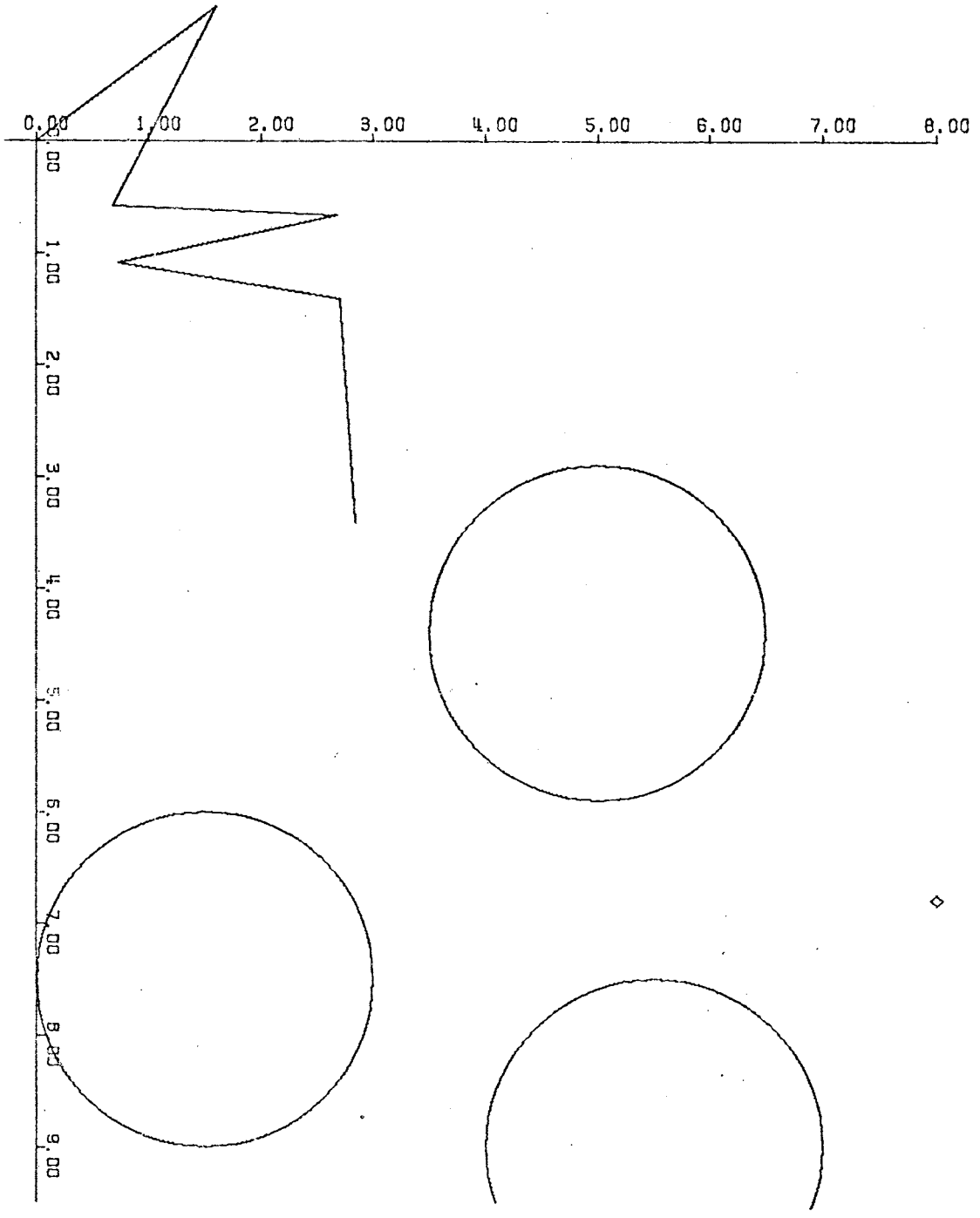

Figure 21. Successive Positions

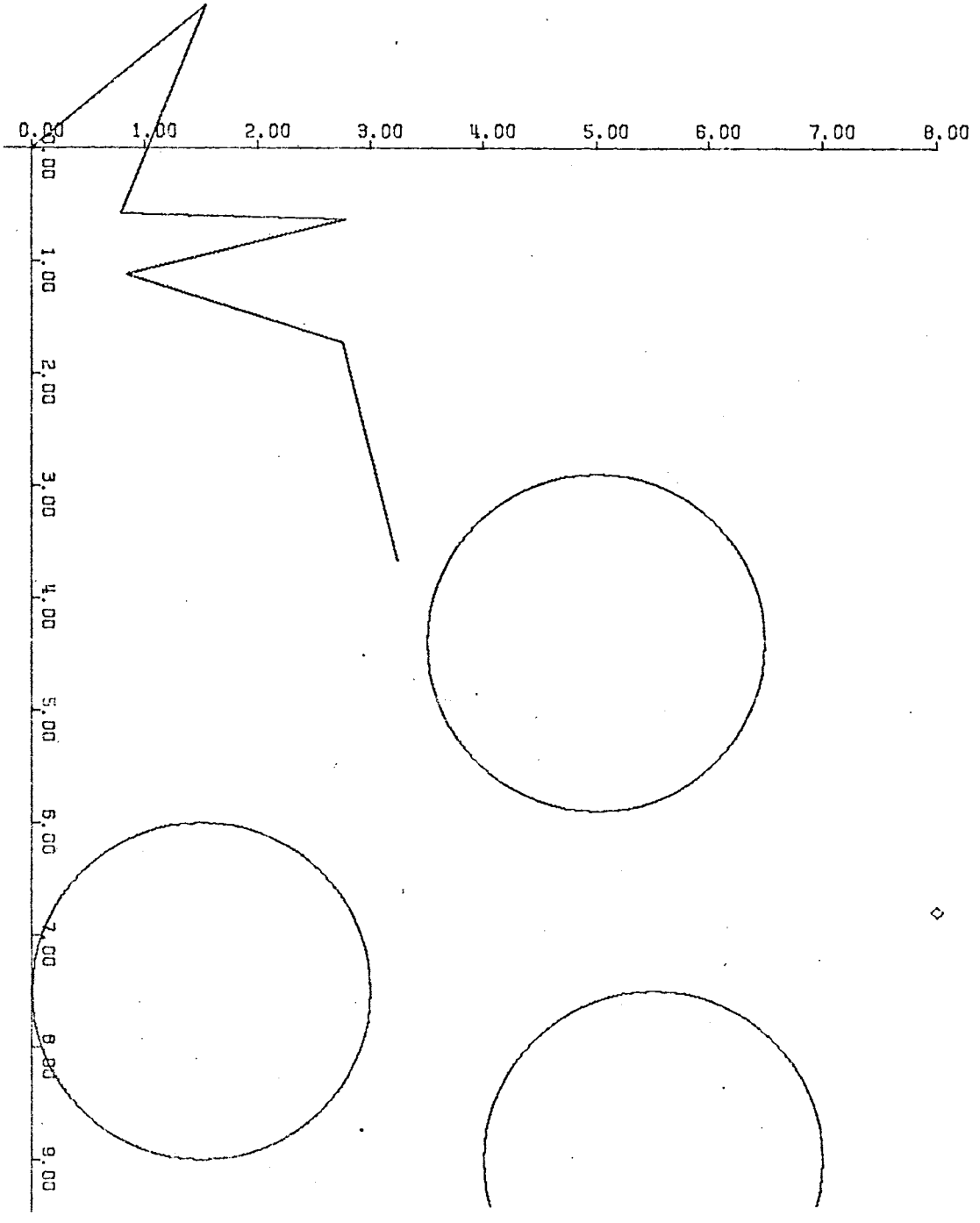

Figure 22. Successive Positions

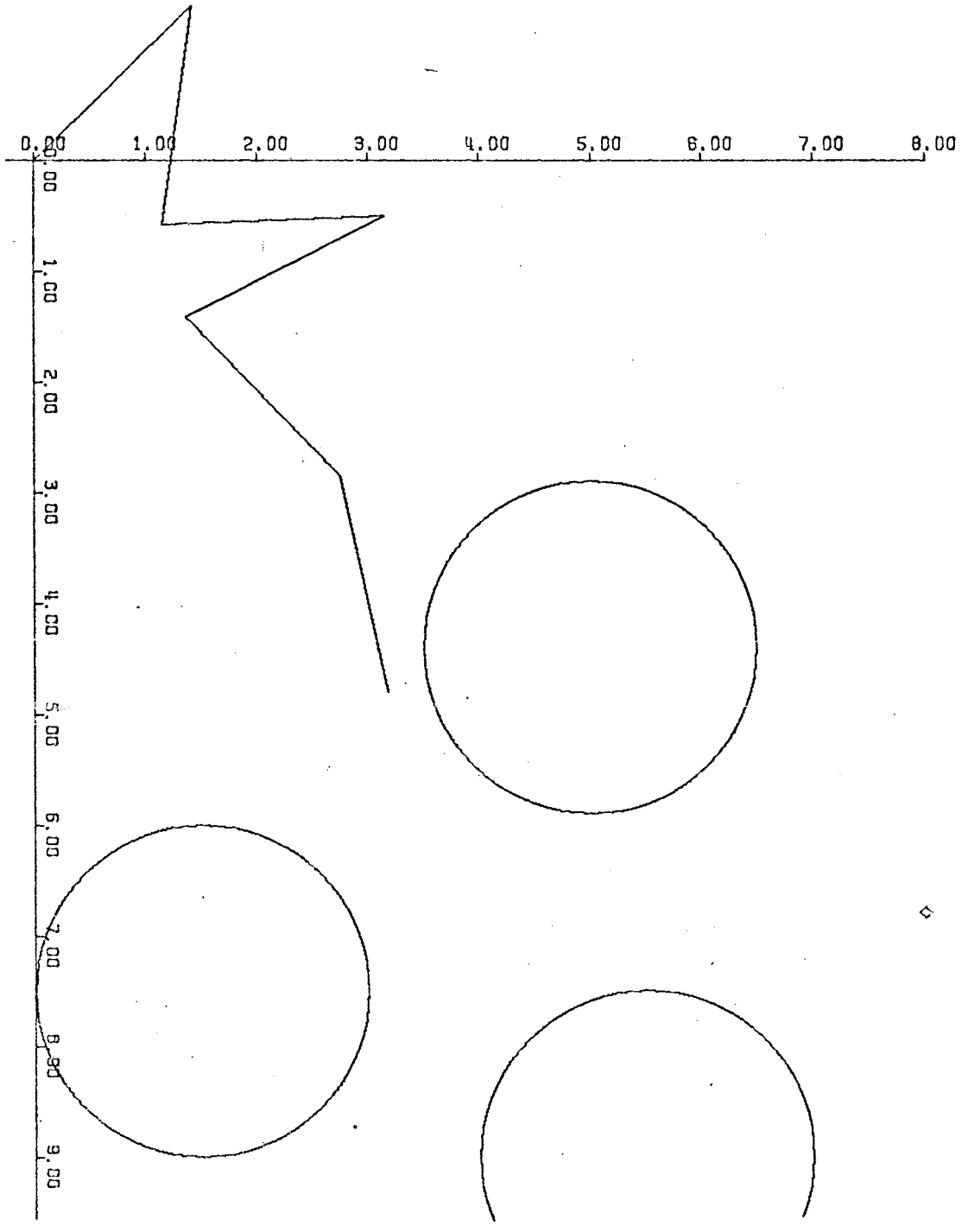

Figure 23. Successive Positions

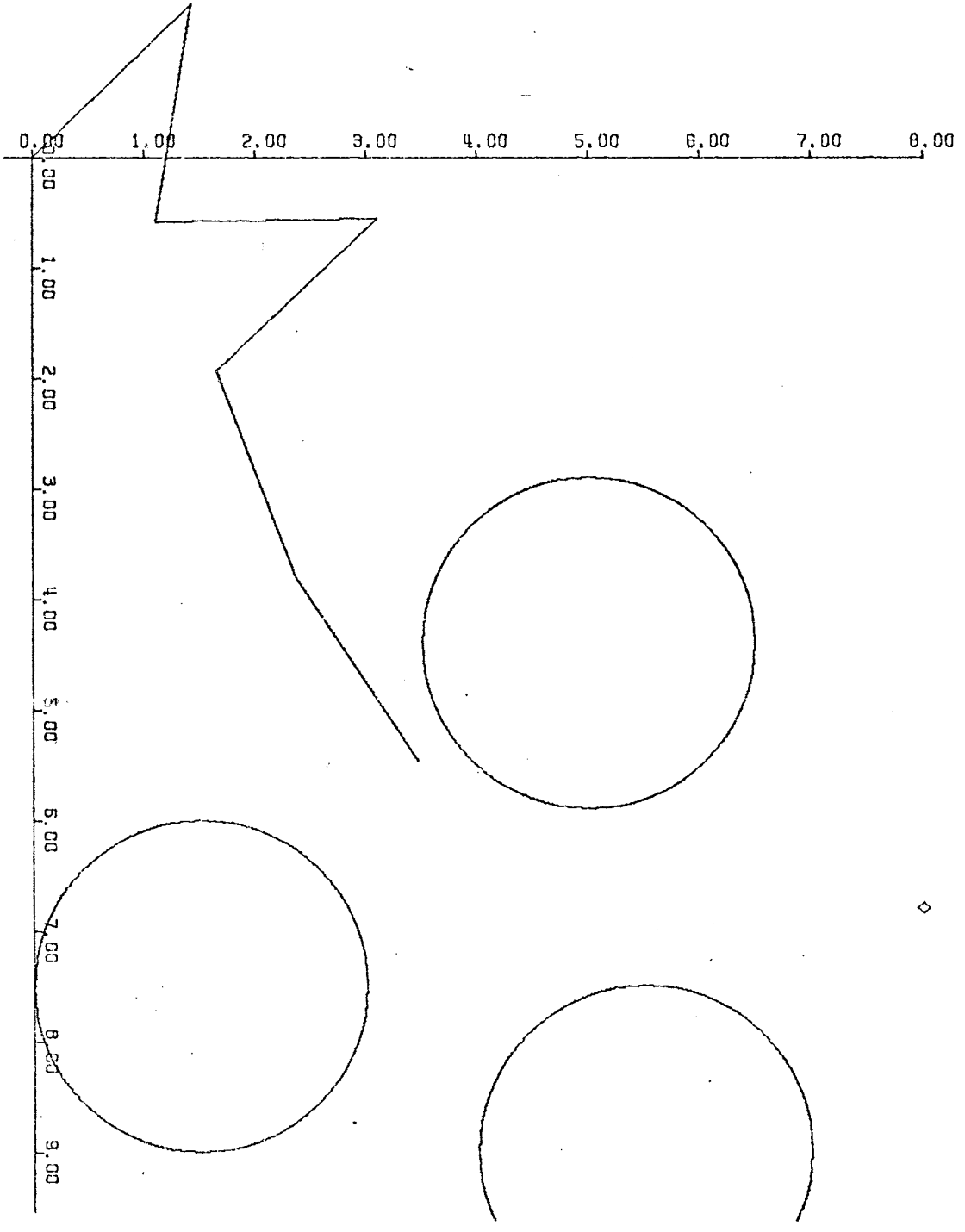

Figure 24. Successive Positions

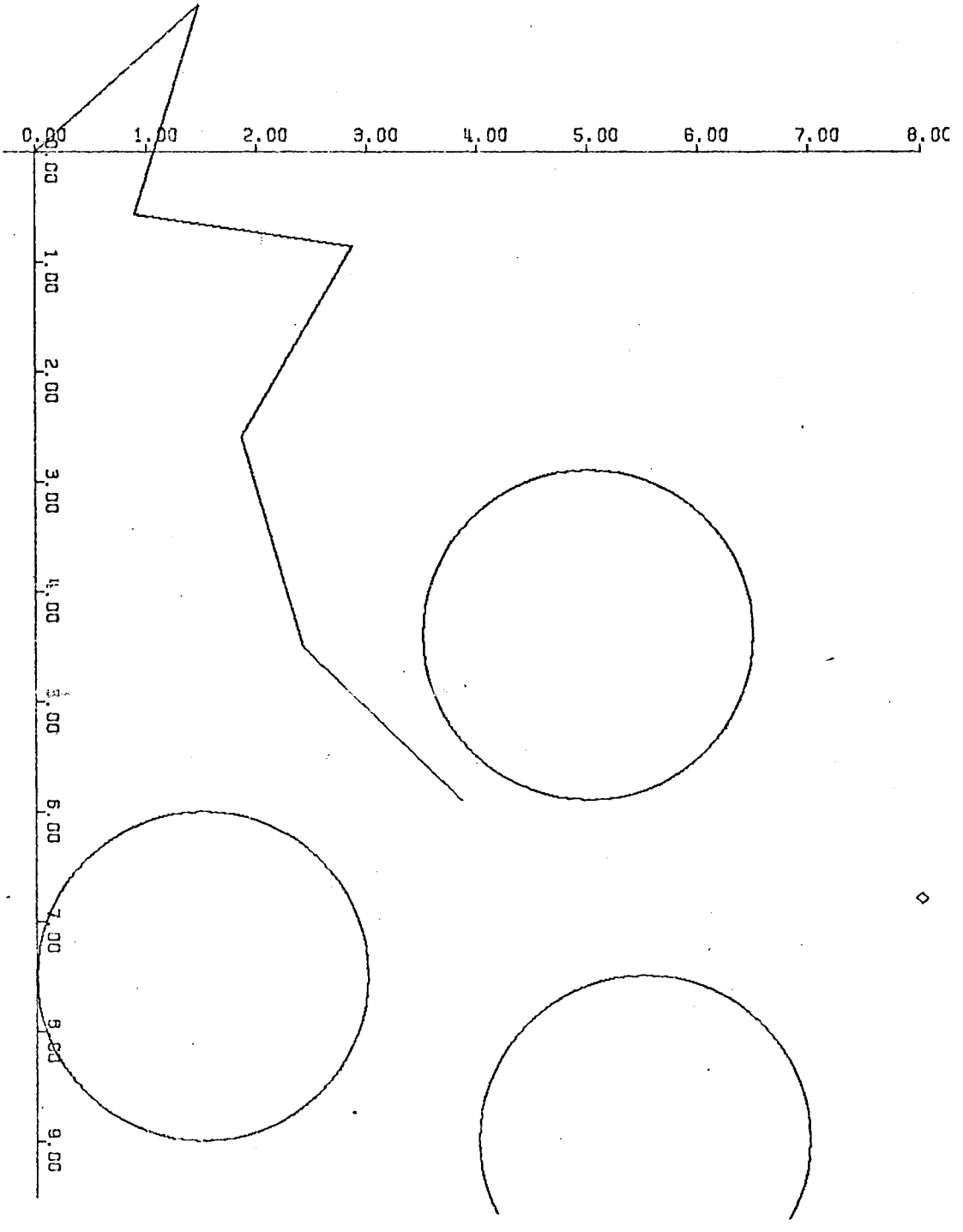

Figure 25. Successive Positions

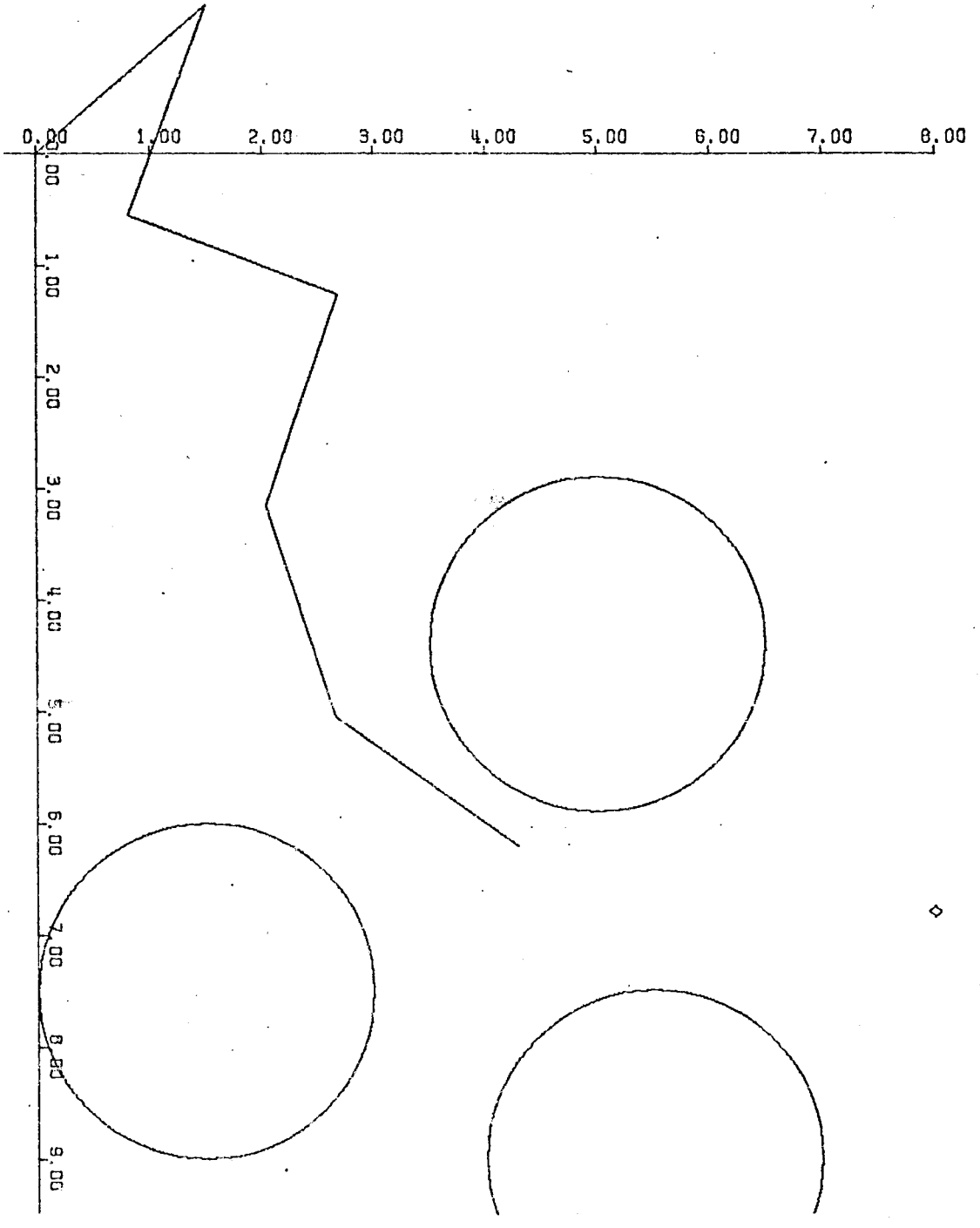

Figure 26. Successive Positions

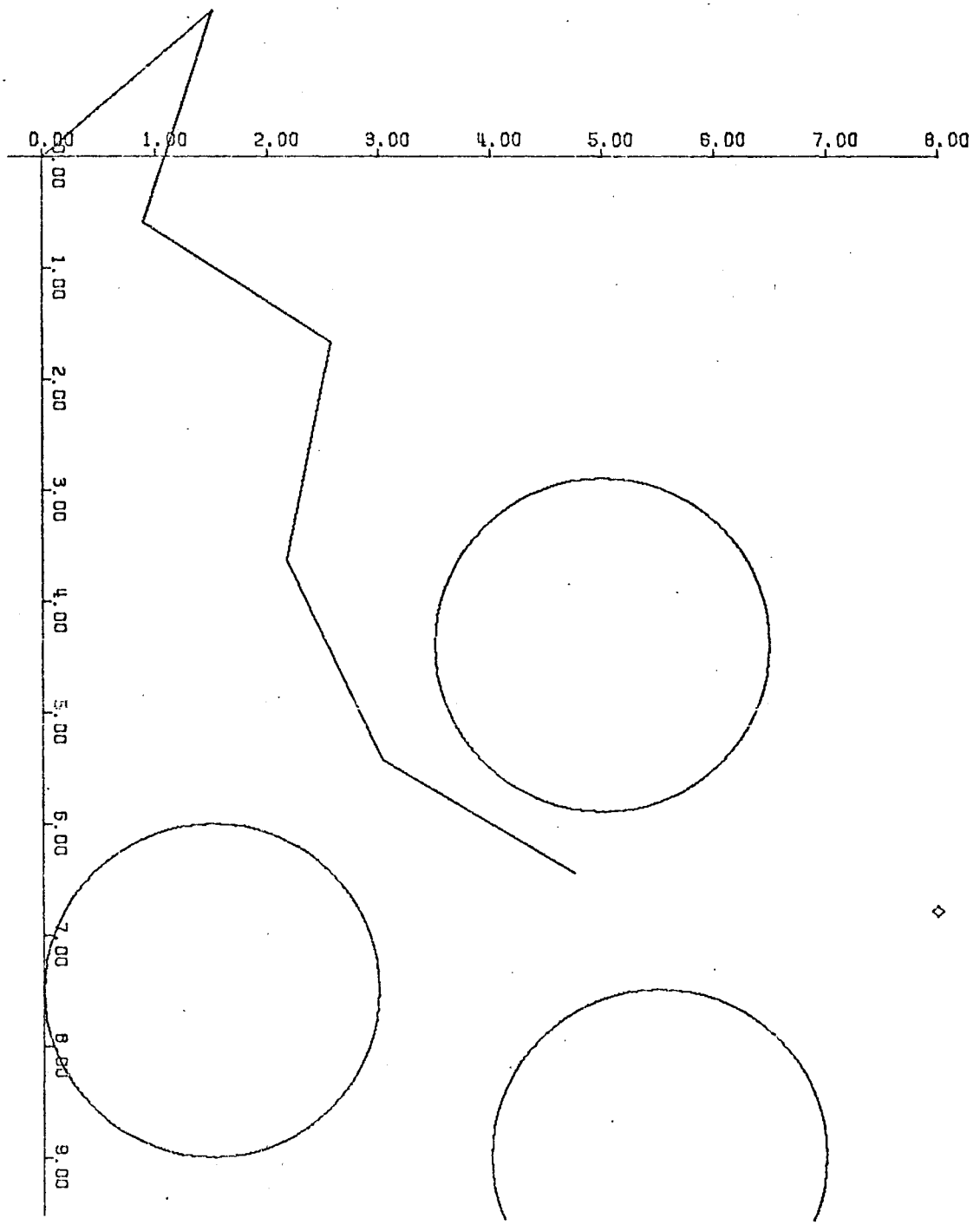

Figure 27. Successive Positions

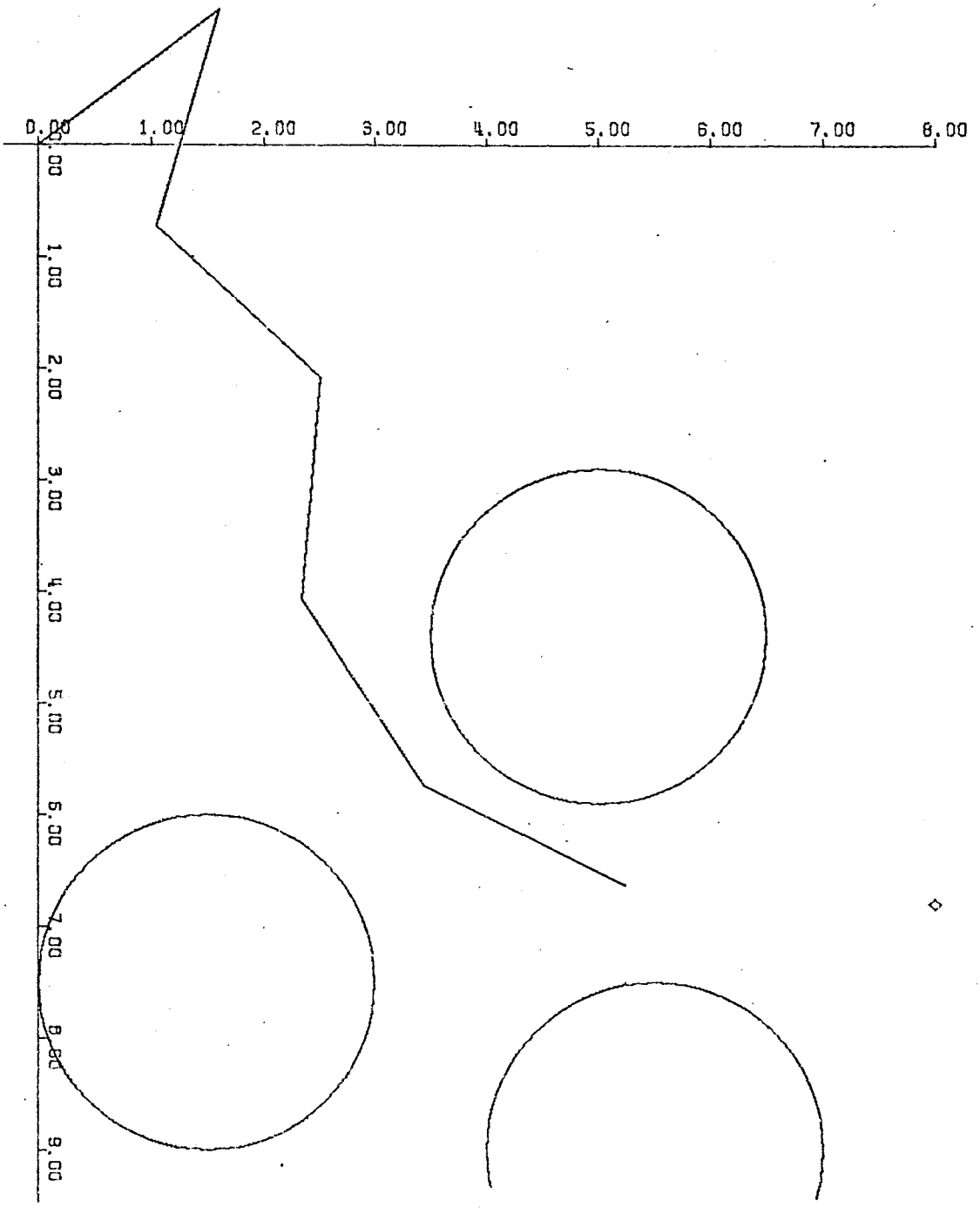

Figure 28. Successive Positions

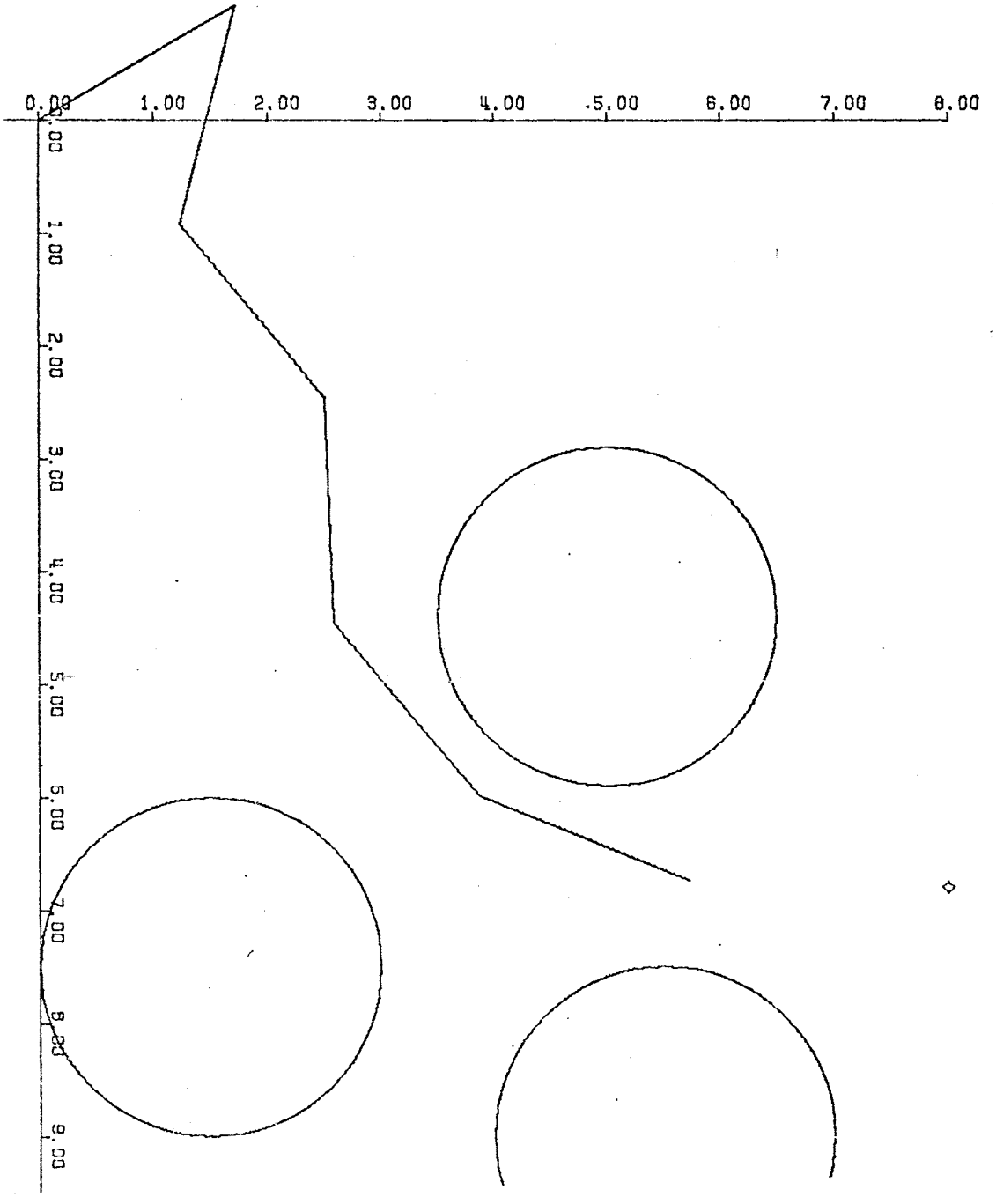

Figure 29. Successive Positions

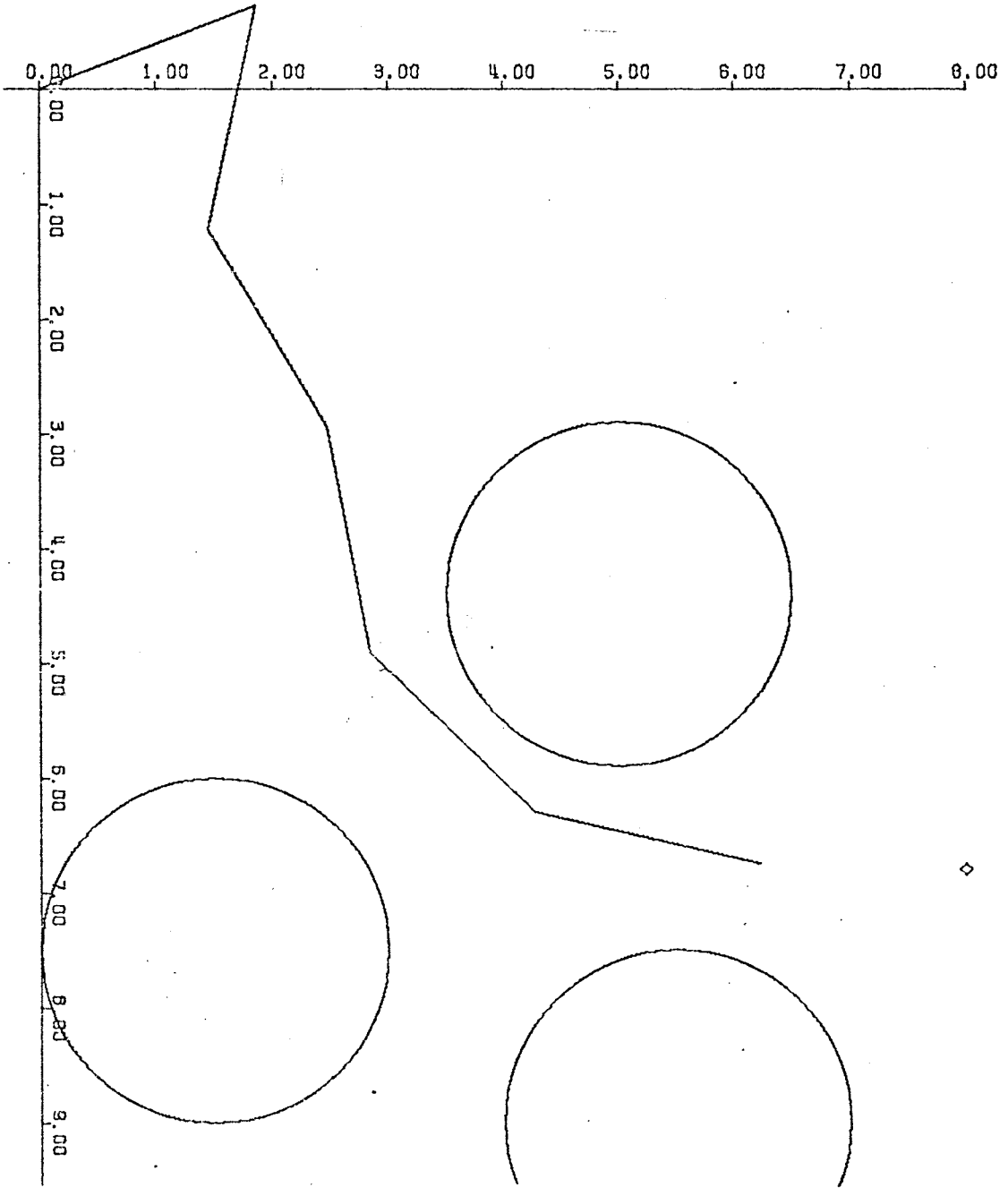

Figure 30. Successive Positions

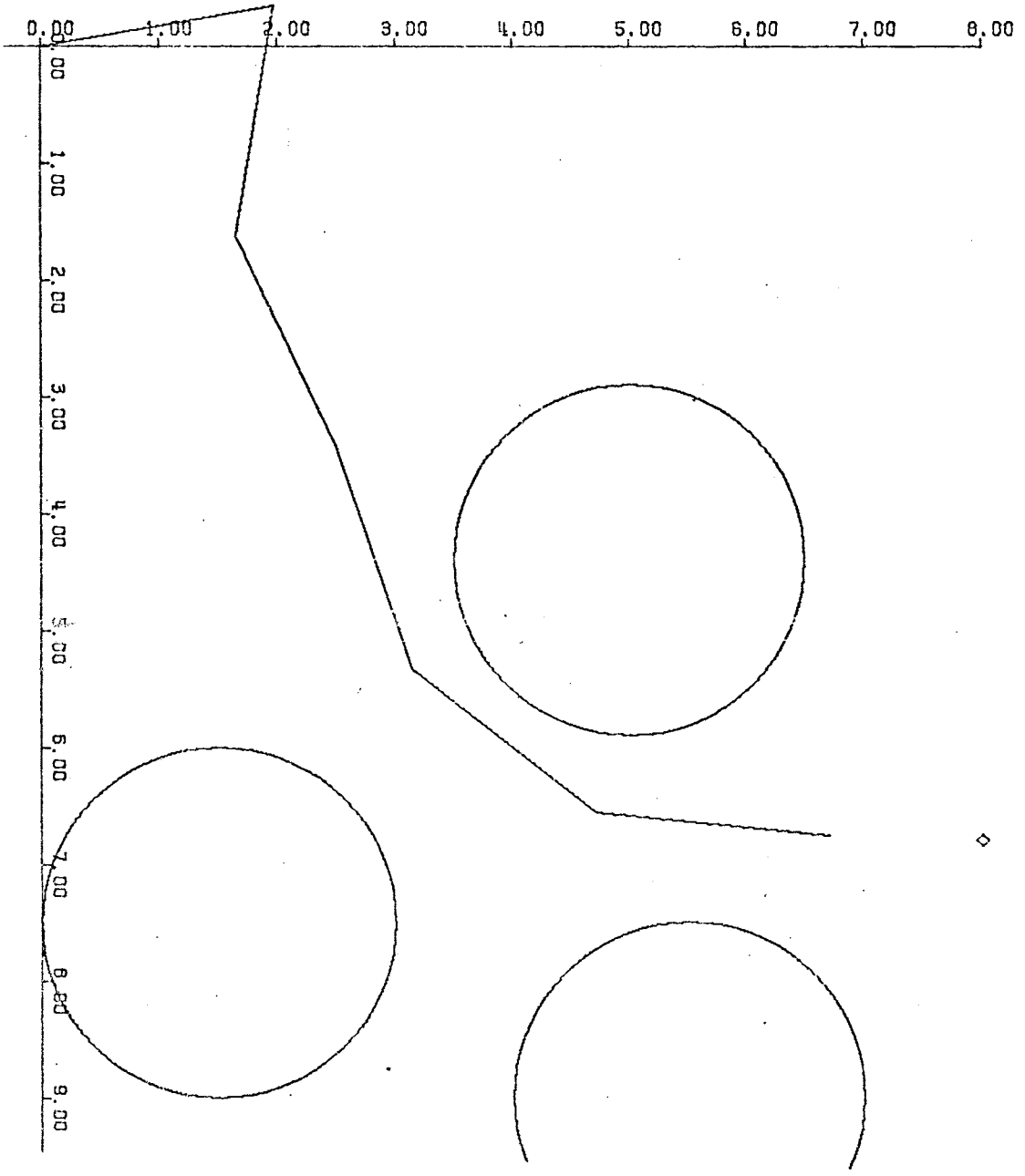

Figure 31. Successive Positions

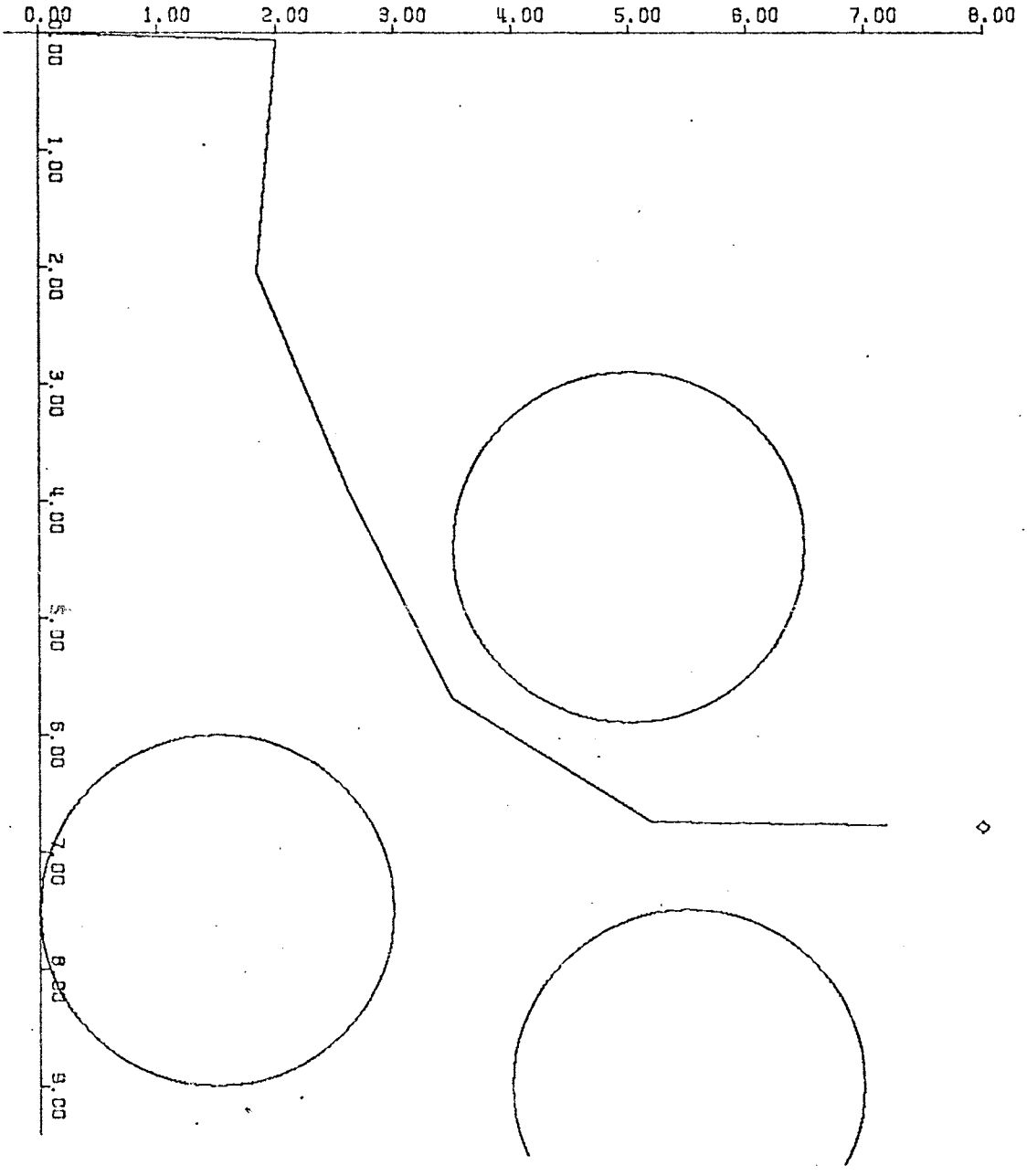

Figure 32. Successive Positions

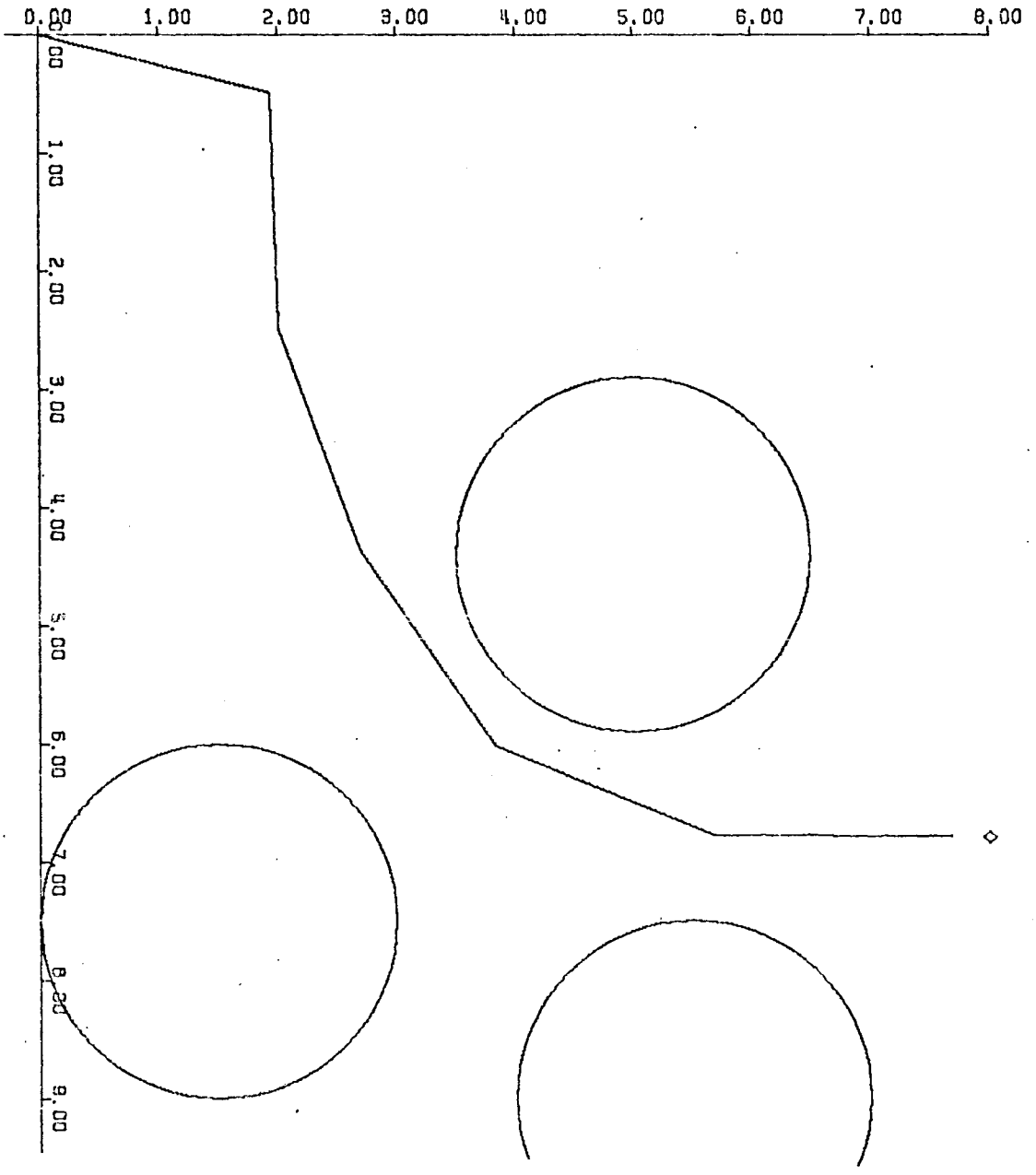

Figure 33. Successive Positions

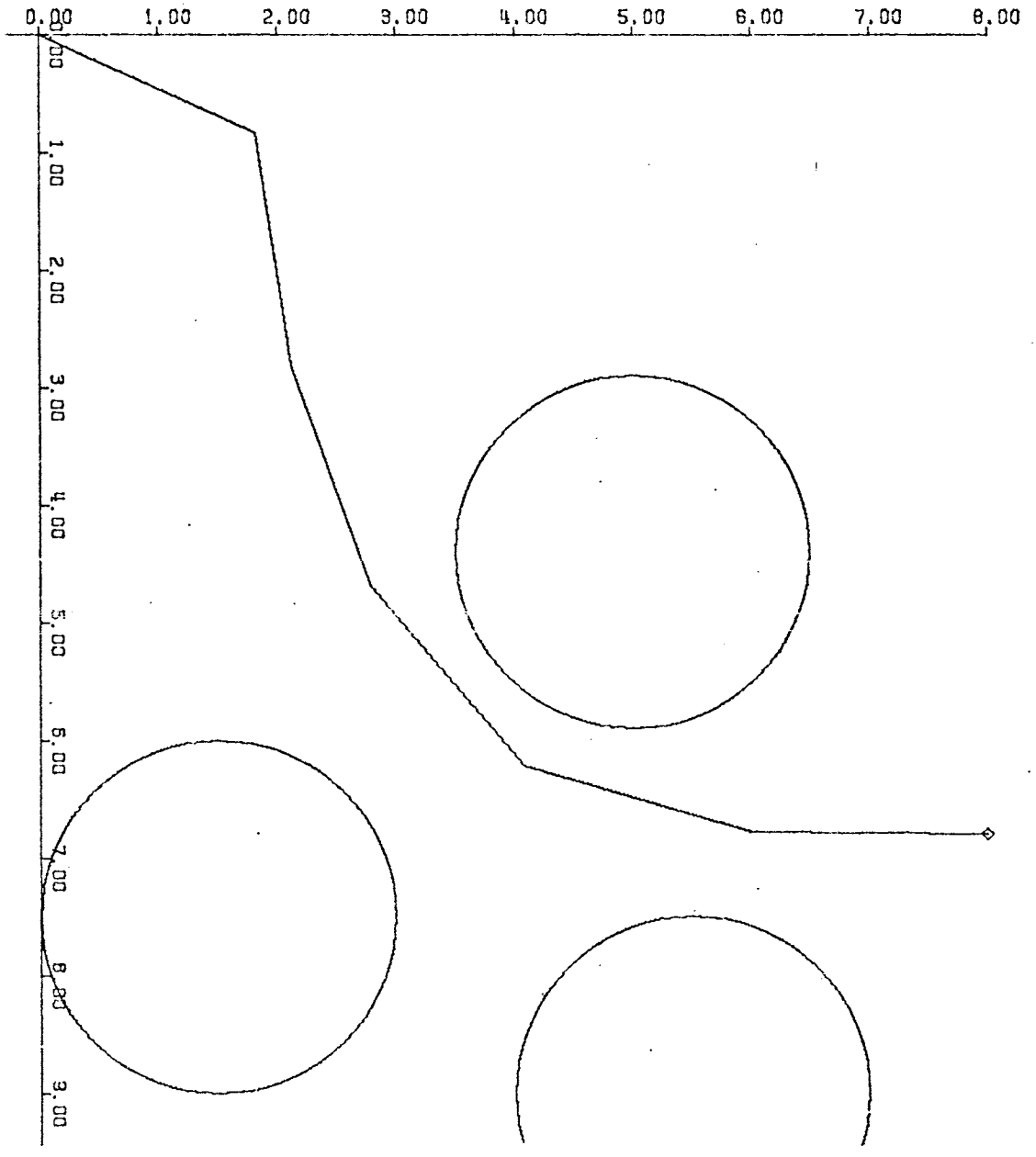

Figure 34. Successive Positions

CHAPTER IV

CONCLUSIONS

The developed algorithm is quite general, can be used with small modifications in any configuration. Its obstacle catalog can be increased with needs. It is fast because only a few square roots besides elementary operations have to be carried out at each cycle, provided trigonometric shaft encoders are used. The square root is a very fast operation in digital computers. It is a non-iterative algorithm and is easy to use. The user has to allocate memory for some arrays, has to specify the upper and lower bounds of the parameters, and has to write a subroutine describing the obstacle configuration. Although the algorithm works on most manipulators and configuration of obstacles, there are some cases for which the algorithm might not be able to proceed past a certain position towards its destination. This is specially true in some three dimensional cases in which the obstacles are closely spaced. This is an inherent limitation of the method which is a sequential process not having learning capability. That is, it cannot improve from previous experience. Another feature which is not implemented, but which could be so if needed is logic decision capabilities, which would allow the selection of a different trajectory if the previously generated one leads to a dead end.

The generated trajectory is one of the infinite available. It is not optimal with respect to any functional, but it fulfills its objec-

tives. Further study needs to be done on trajectory generation in order to select the best approach to the manipulator guidance problem. In this work we have overlooked the dynamic and stress problems in the manipulator. We have also neglected the accuracy problem due to errors in measurement, truncation errors and deflection under loads. All these areas pose an interesting challenge for the future of the manipulator science. The problem of reaching a destination point with a specified attitude of the last link can be easily extended in this algorithm, by adding an influence on the next to the last endpoint. Further work has to be made with respect to how the positions of the obstacles are fed into the digital computer. In our algorithm the position of the obstacles is assumed known a priori, but with further advance in computer science, the positions of the obstacles could be determined be processing the image formed for example in a TV camera or other seeing device.

Finally further work could be done to make the manipulator reach a moving destination point with moving attitude by prediction from the past motion of the destination point and the destination attitude. The prediction would be then used to select the best way to attain its objective.

CHAPTER V

COMPUTER PROGRAMS

Description

The algorithm has been implemented in FORTRAN IV. The part of the program that will be illustrated will be the part that transforms the angles and link lengths into their derivatives with respect to t. This is done by subroutine FCT. The structure of FCT is very similar to the way we subdivided the development of chapter II. The subroutine FCT finds the trigonometric functions of the angles, and then computes step 1) by calling POS. Step 2) is computed by calling the subroutine GUIDE and step 3) is computed by calling VEL. Subroutines POS and VEL make use of subroutines MTG, MTGI, MTGD and SANGI which simply generate all the matrices used by POS and VEL. In addition, MTPL multiplies matrices, and SUBT substracts vectors. Subroutine GUIDE computes ($\dot{P}_1$,... .., $\dot{P}_{N+1}$) by taking care of the length compatibility conditions, and by calling the subroutine OBSTACLE N+1 times (one for each end point) each cycle. Subroutine OBSTACLE computes $\sum_{k} g_{i,k}$ particular type obstacle subroutines and adding the computed influences for all obstacles in space. These particular type obstacle subroutines compute g for different elemental shapes like cylinders, spheres, planes, cones, ellipsoids, toroids, etc. In our simulation the subroutine CYLIN (der) was used as an example of particular type obstacle subroutine. GUIDE uses

MPR for modifying $\dot{P}_2$ as seen at the end of chapter II. In addition OBSTACLE uses SUMU and EQU for performing elementary operations.

Use

In order to use the program, the user should: 1) give the upper and lower bounds of the parameters of the arm ($a_1, \ldots, \theta_N$) by just writing the DATA cards in the subroutine FCT. If the upper and lower bounds are the same then the corresponding parameter will remain fixed throughout the simulation. Give the number of links N according to the definition given in chapter II. 3) Write subroutine OBSTACLE whose job is to compute $\sum_k g_{i,k}$ for each end point of the manipulator, that is for each i. In order to see how this subroutine is written let us refer to table I page 68. In this example three obstacles are used but the procedure for any number of obstacles does not change. The DIMENSION statement has several vectors of dimension four, the G(4), and the GAn(4)'s. Each obstacle has an n associated with it, and there are as many GA's as obstacles. These are used for storing information regarding the influence to the previous link. The JA's are integers which contain information about whether the GA's should be used or not. We also have one JAn for each obstacle. Prior to statement 1 all JA's used should be made equal to minus one. Prior to statement 2 all JA's should be made equal to zero. The rest of the subroutine is a repetition of the following three steps: a) Call a particular type obstacle subroutine. In the example CYLIN is used. This subroutine computes G and J if the points defining the axis of the cylinder XA, YA, ZA and XB, YB, ZB, the radius R, the physical half-size of the link DIST, the end points of the link X1, Y1, Z1, and X2, Y2, Z2, the previous GA and

JA are all given. b) Call EQU which makes GAn for the obstacle equal to the just computed G so that it may be used later. The same happens with JAn and J. c) Call SUMU which adds the vector g to the partial sum $\sum_k g_{i,k}$. By repeating steps a), b) and c) for as many obstacles as there are, the subroutine OBSTACLE is written.

TABLE I

COMPUTER PROGRAM

```
      SUBROUTINE FCT(TIME,THETA,THETD,ALFA,ALFD,A,AD,AY,AYD,AZ,AZD,N)
      DIMENSION THETA(1),THETD(1),ALFA(1),ALFD(1),A(1),AD(1)
      DIMENSION COSTH(6),SINTH(6),COSAL(6),SINAL(6),X(7),Y(7),Z(7),
     1XD(7),YD(7),ZD(7),AP(4,7),CO(7)
          DATA XP,YP,ZP/6.8,8.,0./
      DIMENSION     THMAX(6),THMIN(6),ALMAX(6),ALMIN(6),AMAX(7),AMIN(7)
      DATA AMAX/0.,6*2./,AMIN/0.,6*2./,ALMAX/6*0./,ALMIN/6*0./
      DATA THMAX/6*3.14/,THMIN/6*-3.14/
      DATA AYMAX/0./,AYMIN/0./,AZMAX/0./,AZMIN/0./
       DATA NP/0/
          IF (N.EQ.NP) GO TO 111
           NP=N
          NP1=N+1
  111 CONTINUE
          DO 1 I=1,N
          TE=TAN(ALFA(I)/2.)
          TETE=TE*TE
            DENE=1.+TETE
           SINAL(I)= 2.*TE/DENE
          COSAL(I)=(1.-TETE)/DENE
          T=TAN(THETA(I)/2.)
          TT=T*T
           DENO=1.+TT
          SINTH(I)= 2.*T/DENO
    1   COSTH(I)= (1.-TT)/DENO
              CALL POS(COSTH,SINTH,COSAL,SINAL,A,AY,AZ,X,Y,Z,AP,N)
          CALL MPR(COSTH(1),SINTH(1),COSAL(1),SINAL(1))
              CALL GUIDE( X,Y,Z,XP,YP,ZP,XD,YD,ZD,A,AMAX,AMIN,AY,AYMAX,
     1 AYMIN,AZ,AZMAX,AZMIN,N ,CO     )
              CALL VEL(XD,YD,ZD,COSTH,SINTH,COSAL,SINAL, AYD,AZD   ,
     1 A,AD,THETA,THETD,ALFA,ALFD,THMAX,THMIN,ALMAX,ALMIN,CO,N)
          RETURN
          END

      SUBROUTINE POS(COSTH,SINTH,COSAL,SINAL,A,AY,AZ,X,Y,Z,AP,N)
      DIMENSION COSAL(1),SINAL(1),COSTH(1),SINTH(1),X(1),Y(1),Z(1)
     1, AW(4,4),AP(4,1),A(1)
          DATA NP /0/
          IF(N.EQ.NP) GO TO 111
           NP=N
           NP1=N+1
  111   CONTINUE
             DO      3 I=1,NP1
          K=N-I+1
          AP(1,I)=1.
          AP(2,I)=A(K+1)
          AP(3,I)=0.
          AP(4,I)=0.
       IF(K.EQ.0) GO TO 8
      CALL MTG(COSTH(K),SINTH(K),COSAL(K),SINAL(K),A(K),AW                )
    3   CALL MTPL(AP,AW,AP,4,I,4)
    8     DO 5 J=1,NP1
          X(N+2-J          )=AP(2,J)
         Y(N+2-J)=AP(3,J)    +AY
    5   Z(N+2-J)=AP(4,J)         +AZ
          RETURN
          END
```

TABLE I (Continued)

```
      SUBROUTINE VEL(XD,YD,ZD,COSTH,SINTH,COSAL,SINAL, AYD,AZD    ,
     1 A,AD,THETA,THETD,ALFA,ALFD,THMAX,THMIN,ALMAX,ALMIN,CO,N)
      DIMENSION XD(1),YD(1),ZD(1),A(1),AD(1),THETA(1),THETD(1), CO(1),
     1 ALFA(1),ALFD(1),COSTH(1),SINTH(1),COSAL(1),SINAL(1),ALMAX(1),
     2 ALMIN(1),THMAX(1),THMIN(1),RES(3),VELO(4),POSI(4),PAS(4),PASS(4)
         DIMENSION R(4,4),RI(4,4),RD(4,4),SI(3,4),W(4,4),WI(4,4),WG(4,4)
      DIMENSION W1(4,4),W2(4,4),W3(4,4),W4(4,4),W5(4,4)
C
C     DIMENSION .....,WM(4,4),.........WNM1(4,4)
C
         AD(1)=XD(1)
          AYD=YD(1)
         AZD=ZD(1)
         DO 1  I=1,4
         DO 1  J=1,4
         W(I,J)=0.
         WI(I,J)=0.
    1  WG(I,J)=0.
         DO 2  I=1,4
         W(I,I)=1.
                        WI(I,I)=1.
    2    WG(I,I)=1.
         DO 9  I=1,N
         VELO(1)=0.
         VELO(2)=XD(I+1)
         VELO      (3)=YD(I+1)     -YD(1)
         VELO(4)=ZD(I+1)        -ZD(1)
         POSI(1)=1.
          POSI(2)=A(     I+1)
          POSI(3)=0.
          POSI(4)=0.
         PAS(1)=            0.
         PAS           (2)=AD(I)
         PAS(3)=0.
         PAS(4)=0.
         IF(I.LT.2) GO TO 10
         K=I-1
          IF(I.LT.3) GO TO 11
         L=I-2
      CALL MTG(COSTH(L),SINTH(L),COSAL(L),SINAL(L),A(L), R)
         CALL MTPL(WG,WG,R,4,4,4)
   11 CALL MTGD(COSTH(K),SINTH(K),COSAL(K),SINAL(K),AD(K),ALFD(K),THETD
     1 (K), RD                                                        )
         GO TO      (4,5,6,7,8),K
C
C        GO TO (4,5,6,7,8,.........,15,.........),K
C
     4 CALL MTPL(W1,WG,RD,4,4,4)
         GO TO 3
     5 CALL MTPL(W2,WG,RD,4,4,4)
         GO TO 3
     6  CALL MTPL(W3,WG,RD,4,4,4)
         GO TO 3
     7  CALL MTPL(W4,WG,RD,4,4,4)
         GO TO 3
     8 CALL MTPL(W5,WG,RD,4,4,4)
```

TABLE I (Continued)

```
C
C          ..............
C
C   15 CALL MTPL(WM,WG,RD,4,4,4)
C         GO TO 3
C
C          ..............
C
     3 CONTINUE
        CALL MTGI( COSTH(K),SINTH(K),COSAL(K),SINAL(K),A(K), RI)
        CALL MTG(COSTH(I),SINTH(I),COSAL(I),SINAL(I),A(I),   R)
        CALL MTPL(WI,RI,WI,4,4,4)
       CALL MTPL(WI,W1,R,4,4,4)
        CALL MTPL(PASS,W1,POSI,4,1,4)
        CALL SUBT(VELO,VELO,PASS,4)
        IF(I.LT.3) GO TO 10
        CALL MTPL(W2,W2,R,4,4,4)
         CALL MTPL(PASS,W2,POSI,4,1,4)
         CALL SUBT(VELO,VELO,PASS,4)
         IF       (I.LT.4) GO TO 10
       CALL MTPL(W3,W3,R,4,4,4)
         CALL MTPL(PASS,W3,POSI,4,1,4)
         CALL SUBT(VELO,VELO,PASS,4)
        IF( I.LT.5) GO TO 10
         CALL MTPL(W4,W4,R,4,4,4)
        CALL MTPL(PASS,W4,POSI,4,1,4)
         CALL SUBT(VELO,VELO,PASS,4)
         IF(I.LT.6) GO TO 10
         CALL MTPL(W5,W5,R,4,4,4)
         CALL MTPL(PASS,W5,POSI,4,1,4)
         CALL SUBT(VELO,VELO,PASS,4)
C
C          ..............................................
C
C         IF(I.LT.M+1)       GO TO 10
C         CALL MTPL(WM,WM,R,4,4,4)
C       CALL MTPL(PASS,WM,POSI,4,1,4)
C         CALL   SUBT(VELO,VELO,PASS,4)
C
C          ..............................................
C
   10 CONTINUE
         CALL MTPL(PASS,WI, VELO,4,1,4)
         CALL SUBT(PASS,PASS,PAS,4)
         CALL SANGI(COSTH(I),SINTH(I),COSAL(I),SINAL(I),A(I+1),SI)
         CALL MTPL(RES,SI,PASS,3,1,4)
         AD(I+1)=CO(I+1)*RES(1)
         THETD(I)= RES(2)
         ALFD(I)= RES(3)
      IF((THETA(I).LE.THMIN(I)).AND.(THETD(I).LT.0.)) THETD(I)=0.
      IF((THETA(I).GE.THMAX(I)).AND.(THETD(I).GT.0.)) THETD(I)=0.
      IF((ALFA(I).GE.ALMAX(I)).AND.(ALFD(I).GT.0.)) ALFD(I)=0.
    9 IF((ALFA(I).LE.ALMIN(I)).AND.(ALFD(I).LT.0.)) ALFD(I)=0.
         RETURN
         END
```

TABLE I (Continued)

```
      SUBROUTINE GUIDE (X,Y,Z,XP,YP,ZP,XD,YD,ZD,A,AMAX,AMIN,AY,AYMAX,
     1 AYMIN,AZ,AZMAX,AZMIN,N ,CO     )
         DIMENSION          WD(3)
         DIMENSION        X(7),Y(7),Z(7),XD(7),YD(7),ZD(7),
     1   A(7),AMAX(7),AMIN(7),CO  (   7)
        COMMON/AG/XMPR(3,3)
       DATA DUM/0.4/
         DATA CT/2./
         DATA NP/0/
         IF (N.EQ.NP) GO TO 111
           NP=N
         NP1=N+1
         NP2=N+2
         NP3=N+3
  111  CONTINUE
         DO 4 I=1,NP1
         IF(I.EQ.NP1) GO  TO 5
       CALL OBSTACLE(X(NP2-I),Y(NP2-I),Z(NP2-I),X(NP1-I),Y(NP1-I),Z(NP1-I),
      1),XD(NP2-I),YD(NP2-I),ZD(NP2-I),I )
          GO TO 6
     5 CALL OBSTACLE(X(1),Y(1),Z(1),(X(1)-1.),Y(1),Z(1),XD(1),YD(1),ZD(1),
      1,NP1)
     6                IF(I.GT.1)GO TO 1
       DENN=SQRT((XP-X(NP1))**2+(YP-Y(NP1))**2+(ZP-Z(NP1))**2)
          VX= (XP-X(NP1))/DENN
          VY= (YP-Y(NP1))/DENN
          VZ= (ZP-Z(NP1))/DENN
          PR=XD(NP1)*VX+YD(NP1)*VY+ZD(NP1)*VZ
          VS=DUM-PR
          IF(VS.LT.0.) VS=0.
          IF(VS.GT.(CT*DUM))      VS=CT*DUM
          XD(NP1)=XD(NP1)+VS*VX
          YD(NP1)=YD(NP1)+VS*VY
          ZD(NP1)=ZD(NP1)+VS*VZ
     1 CONTINUE
          IF(I.LE.1) GO TO 2
          VELCT=XD(NP2-I)*UX+YD(NP2-I)*UY+ZD(NP2-I)*UZ
           VSUM=VELPR         -VELCT
       IF(((A(NP3-I).LT.AMAX(NP3-I)).OR.(VSUM.LT.0.)).AND.((A(NP3-I).GT.AMIN(NP3-
      1MIN(NP3-I)).OR.(VSUM.GT.0.)))  VSUM=0.
          CO(NP3-I)=1.
          IF(VSUM.NE.0.) CO(NP3-I)=0.
          XD(NP2-I)=XD(NP2-I)+VSUM*UX
          YD(NP2-I)=YD(NP2-I)+VSUM*UY
          ZD(NP2-I)=ZD(NP2-I)+VSUM*UZ
     2 CONTINUE
          IF(I.       EQ.NP1) GO TO 4
       DEN=SQRT ((X(NP2-I)-X(NP1-I))**2+(Y(NP2-I)-Y(NP1-I))**2+(Z(NP2-I)-Z(NP1-I)
      1Z(NP1-I))**2)
          UX=(X(NP2-I)-X(NP1-I))/DEN
          UY=(Y(NP2-I)-Y(NP1-I))/DEN
          UZ=(Z(NP2-I)-Z(NP1-I))/DEN
          VELPR= XD(NP2-I)*UX+YD(NP2-I)*UY+ZD(NP2-I)*UZ
        IF(I.NE.(N-1))    GO TO 4
          VX=UX
          VY=UY
```

TABLE I (Continued)

```
      VZ=UZ
      V3=VELPR
 4 CONTINUE
      IF(A(1).LT.AMAX(1)) GO TO 20
      IF      (XD(1). GE.0.) XD(1)=0.
20 IF(A(1).GT.AMIN(1)) GO TO 21
       IF(XD(1).LE.0.) XD(1)=0.
21     IF(AY.LT.AYMAX) GO TO 22
    IF(YD(1).GE.0.) YD(1)=0.
22  IF(AY.GT.AYMIN) GO TO 23
      IF(YD(1).LE.0.) YD(1)=0.
23  IF(AZ.LT.AZMAX) GO TO 24
      IF(ZD(1).GE.0.) ZD(1)=0.
24 IF(AZ.       GT.AZMIN) GO TO 25
      IF(ZD(1).LE.0.)       ZD(1)=0.
25     CONTINUE
       IF(VSUM.EQ.0.) GO TO 30
     WD(1) = XD(2)-XD(1)
         WD(2)= YD(2)      -YD(1)
       WD(3)= ZD(2)  -ZD(1)
       CALL MTPL(WD,XMPR,WD,3,1,3)
       CTE=V3/ (WD(1)*VX+WD(2)*VY+WD(3)*VZ)
        XD(2)= CTE*WD(1)+ XD(1)
       YD(2)       = CTE*WD(2) +YD(1)
       ZD(2)=CTE*WD(3) +ZD(1)
30 CONTINUE
      RETURN
      END



      SUBROUTINE OBSTACLE(X1,Y1,Z1,X2,Y2,Z2,XD,YD,ZD,I        )
      DIMENSION         G(4),GA1(4),GA2(4),GA3(4)
      XD=0.
       YD=0.
       ZD=0.
      IF(I.NE.0) GO TO 1
      JA1=-1
      JA2=-1
       JA3=-1

1 IF(I.NE.1) GO TO 2
      JA1=0
      JA2=0
      JA3=0

2  CALL CYLIN(4.4,5.,-5.,4.4,5.,5.,1.5,0.,X1, Y1,Z1,X2,Y2,Z2,GA1,
 1    JA1,G,J)
       CALL EQU(GA1,G,JA1,J)
       CALL SUMU(XD,YD,ZD,G)
  CALL CYLIN(9.,5.5,-5.,9.,5.5,5.,1.5,0.,X1,Y1,Z1,X2,Y2,Z2,GA2,JA2,
 1 G,J)
       CALL SUMU(XD,YD,ZD,G)
        CALL EQU( GA2,G,JA2,J)
  CALL CYLIN(7.5,1.5,-5.,7.5,1.5,5.,1.5,0.,X1,Y1,Z1,X2,Y2,Z2,GA3,JA3
 1,G,J)
       CALL SUMU(XD,YD,ZD,G)
       CALL EQU(GA3,G,JA3,J)
       RETURN
        END
```

TABLE I (Continued)

```
      SUBROUTINE CYLIN (XA,YA,ZA,XB,YB,ZB,R,DIST,X1,Y1,Z1,X2,Y2,Z2,GA,JA,
     1 A,G,J)
        DIMENSION G(4),GA(4)
        DATA N/3/,DED/2./
        DATA H/0.4/
      CS=(XB-XA)**2+(YB-YA)**2+(ZB-ZA)**2
       FS=(X1-XA)*(XB-XA)+(Y1-YA)*(YB-YA)+(Z1-ZA)*(ZB-ZA)
        IF(JA.EQ.(-1)) GO TO 3
      AS=(XB-XA)*(X2-X1)+(YB-YA)*(Y2-Y1)+(ZB-ZA)*(Z2-Z1)
       BS=-(X2-X1)**2-(Y2-Y1)**2-(Z2-Z1)**2
       DS=-AS
       ES=(X1-XA)*(X2-X1)+(Y1-YA)*(Y2-Y1)+(Z1-ZA)*(Z2-Z1)
        DEN= AS*DS-BS*CS
        XL= (ES*DS-BS*FS)/DEN
        XM=  (AS*FS-CS*ES)/DEN
       IF((    XL.LT.1).AND.(XL.GT.0.)) GO TO 2
    4 DO 1 KI=1,4
    1  G(KI)=0.
        RETURN
    2 IF ((XM.LT.0.).OR.(XM.GT.1.))                    GO TO 3
        DELTX= X1-XA+ XM*(X2-X1)- XL*(XB-XA)
        DELTY= Y1-YA+ XM*(Y2-Y1)- XL*(YB-YA)
        DELTZ= Z1-ZA+ XM*(Z2-Z1)- XL*(ZB-ZA)
         D= SQRT( DELTX*DELTX+DELTY*DELTY+DELTZ*DELTZ)
         DE=D*(D-R-DIST)**N
         IF((D-R-DIST).LE.0.) INT=2
         DNUM= 2.*H*(H-D+R+DIST)
          IF(DNUM.LT.0.) DNUM=0.
         G(1)= DNUM*DELTX/DE
         G(2)= DNUM*DELTY/DE
         G(3)= DNUM*DELTZ/DE
         G(4)= DNUM*D/DE
         J=1
         IF(JA.EQ.0) RETURN
      G(4)=SQRT(G(4)*G(4)+GA(4)*GA(4)+2.*(G(1)*GA(1)+G(2)*GA(2)+G(3)*
     1 GA(3)))  /DED
         G(1)=(G(1)+GA(1))/DED
         G(2)=(G(2)+GA(2))/DED
         G(3)=(G(3)+GA(3))/DED
         RETURN
    3  XL=FS/CS
          J=0
      IF(XM.GT.1.) GO TO 4
         DELTX= X1-XA -XL*(XB-XA)
         DELTY= Y1-YA -XL*(YB-YA)
         DELTZ= Z1-ZA -XL*(ZB-ZA)
         D= SQRT(DELTX*DELTX+DELTY*DELTY+DELTZ*DELTZ)
        IF((D-R-DIST).GT.0.) GO TO 18
         IF(INT.EQ.2) GO TO 18
          INT=1
   18  CONTINUE
         DE=D*(D-R-DIST)**N
         DNUM= 2.*H*(H-D+R+DIST)
          IF(DNUM.LT.0.) DNUM=0.
         G(1)= DNUM*DELTX/DE
         G(2)= DNUM*DELTY/DE
         G(3)= DNUM*DELTZ/DE
        G(4)= DNUM*D/DE
         IF(JA.NE.1) RETURN
         IF(G(4).GE.GA(4))  RETURN
       DO 5 II=1,4
    5   G(II)=GA(II)
          RETURN
         END
```

TABLE I (Continued)

```
      SUBROUTINE MTG(COSTH,SINTH,COSAL,SINAL,A, AZ        )
      DIMENSION         AZ(4,4)
       AZ(1,1)              =1.
       AZ(2,1)=A
       AZ(3,1)=0.
       AZ(4,1)=0.
       AZ(1,2)=0.
     AZ(2,2)=COSTH
        AZ(3,2)=  COSAL*SINTH
       AZ(4,2)= SINAL*SINTH
       AZ(1,3)= 0.
        AZ(2,3)=  -SINTH
       AZ(3,3)= COSAL*COSTH
        AZ(4,3)= SINAL*COSTH
       AZ(1,4)=  0.
       AZ(2,4)=0.
       AZ(3,4)=        -SINAL
       AZ(4,4)= COSAL
       RETURN
       END
       SUBROUTINE MTGI(COSTH,SINTH,COSAL,SINAL,A,  AZ)
     DIMENSION           AZ(4,4)
        AZ(1,1)= 1.
        AZ(2,1)=         -A*COSTH
        AZ(3,1)= A*SINTH
        AZ(4,1)= 0.
         AZ(1,2)=0.
        AZ(2,2)=  COSTH
        AZ(3,2)= -SINTH
        AZ(4,2)=0.
        AZ(1,3)=0.
                AZ(2,3)= COSAL*SINTH
      AZ(3,3)= COSAL*COSTH
        AZ(4,3)=-SINAL
        AZ(1,4)=0.
        AZ(2,4)=     SINAL*SINTH
        AZ(3,4)= SINAL*COSTH
         AZ(4,4)=  COSAL
         RETURN
        END


       SUBROUTINE MTGD(COSTH,SINTH,COSAL,SINAL, AD,ALFD,THETD,AZ)
      DIMENSION         AZ(4,4)
          DO 1 I=1,4
     1    AZ      (1,I)=0.
       AZ(2,1)=AD
         AZ(3,1)=0.
          AZ(4,1)=0.
           AZ(2,2)=-THETD*SINTH
        AZ(3,2)=-ALFD*SINAL*SINTH+THETD*COSAL*COSTH
       AZ(4,2)= ALFD*COSAL*SINTH+THETD*SINAL*COSTH
         AZ(2,3)=-THETD*COSTH
        AZ(3,3)= -ALFD*SINAL*COSTH-THETD*COSAL*SINTH
        AZ(4,3)= ALFD*COSAL*COSTH-THETD*SINAL*SINTH
         AZ(2,4)=0.
         AZ(3,4) =     -ALFD*COSAL
         AZ(4,4)=       -ALFD*SINAL
          RETURN
          END
```

TABLE I (Continued)

```
      SUBROUTINE SANGI  (COSTH,SINTH,COSAL,SINAL,A, SI          )
      DIMENSION         SI( 3,4)
      SI(1 ,1 )=0.
      SI(2,1)=0.
      SI(3,1)=0.
       SI(1 ,2) = COSTH
       SI( 1,3) =SINTH*COSAL
      SI(1,4)= SINTH*SINAL
      SI(2,2)=-SINTH/A
       SI( 2,3)= COSAL*COSTH/A
      SI(2,4)=  SINAL*COSTH/A
      SI(3,2)=0.
      SI(3,3)= -SINAL/(A*SINTH)
      SI(3,4)=  COSAL/(A*SINTH)
      RETURN
      END
      SUBROUTINE MPR(COSTH,SINTH,COSAL,SINAL)
        COMMON/AG/XMPR(3,3)
         XMPR(1,1)= SINTH*SINTH
         XMPR(2,1)=-COSTH*SINTH*COSAL
         XMPR(3,1)= -COSTH*SINTH*SINAL
          XMPR(2,3)  =      -SINTH*SINTH*COSAL*SINAL
        XMPR(2,2)= 1.-SINTH*SINTH*COSAL*COSAL
        XMPR(3,3)= 1.-SINTH*SINTH*SINAL*SINAL
         XMPR(1,2)=XMPR(2,1)
          XMPR(1,3)=XMPR(3,1)
         XMPR(3,2)=XMPR(2,3)
         RETURN
         END
       SUBROUTINE SUBT(U,V,W,N)
      DIMENSION           U(N),V(N),W(N)       ,AU(8)
         DO 1 I=1,N
    1  AU(I)=V(I)-W(I)
         DO 2 I=1,N
    2  U(I)=AU(I)
          RETURN
           END
       SUBROUTINE SUMU(XD,YD,ZD,G)
       DIMENSION G(4)
        XD=XD+G(1)
              YD=YD+G(2)
        ZD=ZD+G(3)
      RETURN
       END
          SUBROUTINE EQU(GA,G,JA,J)
           DIMENSION        G(4),GA(4)
           DO 1 JJ=1,4
    1     GA(JJ)=G(JJ)
           JA=J
          RETURN
           END
          SUBROUTINE MTPL(C,A,B,N,M,L)
C            C=A*B
         DIMENSION          C(N,M),A(N,L),B(L,M), D(8,8)
          DO 1 I=1,N
          DO 1 J=1,M
             D(I,J)=0.
             DO 1 K=1,L
        1  D(I,J)=D(I,J)+A(I,K)*B(K,J)
             DO 2 I=1,N
             DO 2 J=1,M
        2 C(I,J)=D(I,J)
             RETURN
             END
```

BIBLIOGRAPHY

1. Ernst, H. A., "A Computer-Controlled Mechanical Hand", (Sc.D. Thesis, M.I.T., Dept. of Electrical Engineering, 1961).

2. Rosen, C. A., and N. J. Nilsson, "An Intelligent Automaton", IEEE International Convention Record, 1967.

3. Beckett, J. T., "A Computer-Aided Control Technique for a Remote Manipulator", Digital Systems Laboratory, Case Institute of Technology, paper no. 1-67-44, 1967.

4. Hartenberg, R. S., and J. Denavit, Kinematic Synthesis of Linkages, New York: McGraw-Hill, 1964.

5. Beggs, J. S., Advanced Mechanisms, New York: MacMillan, 1966.

6. Whitney, D. E., "State Space Models of Remote Manipulation Tasks", (PhD Thesis, M.I.T., Dept. of Mechanical Engineering, January, 1968).

7. Pieper, D. L., "The Kinematics of Manipulators under Computer Control", (PhD Thesis, Computer Science Dept., Stanford University, October, 1968).

APPENDIX

SOME OBSTACLES AND THEIR INFLUENCES

We will develop the procedure by which the vector g is computed for certain elementary obstacles which are, triangular plane elements, spheres and cylinders. Other obstacles could be approximated by combinations of the three mentioned above.

Triangular Plane Element

For a plane or region of plane only an endpoint can be closest to it. Let (a, b, c) be a vector and (x_o, y_o, z_o) be a point in space. The plane that is perpendicular to the vector and passes by the point is,

$$a(x - x_o) + b(y - y_o) + c(z - z_o) = 0 \qquad \text{(A-1)}$$

By analytic geometry the distance d from a point (x_p, y_p, z_p) to the plane (A-1) is

$$d = \frac{a(x_p - x_o) + b(y_p - y_o) + c(z_p - z_o)}{s} \qquad \text{(A-2)}$$

Where $s = \sqrt{a^2 + b^2 + c^2}$ (A-3)

The continuous and decreasing function of d mentioned previously will

be chosen in this and in the next cases to be,

$$f(d) = \frac{2h(h-d)}{d^3} \qquad 0 < d < h \tag{A-4}$$

$$f(d) = 0 \qquad d > h \tag{A-5}$$

and $$g = f(d).(a/s, b/s, c/s) \tag{A-6}$$

where h is an arbitrary distance.

So far we have considered an unlimited plane. Let us now see how to compute g for a triangular plane element. The vector g computed at (A-6) will also hold for a triangular element provided the endpoint is on the outward side of the triangle (this is because the triangle will always part of a solid body) and its projection on the plane of the triangle falls inside it. Let the triangular element be defined by its three vertices (x_a, y_a, z_a), (x_b, y_b, z_b) and (x_c, y_c, z_c) such that the succession a, b, c define an outward going normal when a right hand screw is considered. In this case the components of the normal to the plane when expressed as a function of the coordinates of P_a, P_b and P_c are,

$$a = (y_b-y_a)(z_c-z_a) - (z_b-z_a)(y_c-y_a) \tag{A-7}$$

$$b = (z_b-z_a)(x_c-x_a) - (x_b-x_a)(z_c-z_a) \tag{A-8}$$

$$c = (x_b-x_a)(y_c-y_a) - (y_b-y_a)(x_c-x_a) \tag{A-9}$$

To check whether P_p is in the outward face of the triangle we find

d by using (A-2) and (A-3) and check its sign. If d is positive then point P_p is towards the outward face of the triangle. To check whether the projection of P_p on the plane of the triangle falls inside it, have to check whether the projection of P_p falls in one of the semi-planes defined by each of the sides of the triangle. So for this to happen each and all of these three determinants should be negative,

$$\begin{vmatrix} a & b & c \\ (x_p-x_a) & (y_p-y_a) & (z_p-z_a) \\ (x_b-x_a) & (y_b-y_a) & (z_b-z_a) \end{vmatrix} < 0 \qquad \text{(A-10)}$$

$$\begin{vmatrix} a & b & c \\ (x_p-x_b) & (y_p-y_b) & (z_p-z_b) \\ (x_c-x_b) & (y_c-y_b) & (z_c-z_b) \end{vmatrix} < 0 \qquad \text{(A-11)}$$

$$\begin{vmatrix} a & b & c \\ (x_p-x_c) & (y_p-y_c) & (z_p-z_c) \\ (x_a-x_c) & (y_a-y_c) & (z_a-z_c) \end{vmatrix} < 0 \qquad \text{(A-12)}$$

So if d is positive and the three determinants are negative g will be computed by (A-6). Otherwise, g will be taken as zero, this meaning that the face of the triangle itself has no influence although the sides and/or the vertices might have.

Sphere

Let the center of the sphere be P_c (x_c, y_c, z_c), the radius be R (for a point R equals zero) and let a link be given by its two end-

points P_1 (x_1, y_1, z_1) and P_2 (x_2, y_2, z_2). In order to find if the endpoints or the segment itself are closest to P_c, we project P_c on the line defined by P_1 and P_2; the equation of the line is,

$$\begin{aligned} x &= x_1 + \mu(x_2-x_1) \\ y &= y_1 + \mu(y_2-y_1) \\ z &= z_1 + \mu(z_2-z_1) \end{aligned} \qquad \text{(A-13)}$$

The value of the parameter will determine which endpoint or segment will be closest to P_c , by minimizing the square of the distance from P_c to $P(\mu)$. The result is,

$$\mu = \frac{(x_c-x_1)(x_2-x_1) + (y_c-y_1)(y_2-y_1) + (z_c-z_1)(z_2-z_1)}{(x_2-x_1)^2 + (y_2-y_1)^2 + (z_2-z_1)^2} \qquad \text{(A-14)}$$

If $\mu<0$, P_1 is closest to P_c. If $0<\mu<1$ the segment P_1P_2 is closest to P_c. If $\mu>1$ P_2 is closest to P_c. As we know μ we know $P(\mu)$, we can find the vector $(P_c-P(\mu))$ and its magnitude which is the distance d. If $\mu<0$ we have to take $\mu=0$ because P_1 is then the nearest point to P_c. Similarly if $\mu>1$ we have to take $\mu=1$. We have now all the data for computing g.

Cylinder

Let the cylinder be defined by its axis and its radius. Let the axis be defined by two points P_a and P_b. We are interested only in limited cylinders, that is the influence will be zero if it falls outside the segment defined by P_a and P_b. Let the endpoints of a link be P_1 and P_2. Then the equations of the axis and of the link are,

$$x_p = x_1 + \mu(x_2 - x_1)$$
$$y_p = y_1 + \mu(y_2 - y_1) \qquad \text{(A-15)}$$
$$z_p = z_1 + \mu(z_2 - z_1)$$

$$x_{ax} = x_a + \lambda(x_b - x_a)$$
$$y_{ax} = y_a + \lambda(y_b - y_a) \qquad \text{(A-16)}$$
$$z_{ax} = z_a + \lambda(z_b - z_a)$$

To find the influence of the cylinder on the link we find the common normal. This we do by minimizing the square of the distance $P_{ax}P_p$ with respect to the parameters μ and λ. This gives,

$$\lambda = \frac{d_s e_s + b_s f_s}{a_s d_s + b_s c_s} \quad \text{and} \quad \mu = \frac{a_s f_s - c_s e_s}{a_s d_s + b_s c_s} \qquad \text{(A-17-18)}$$

$$a_s = (x - x_a)(x_2 - x_1) + (y_b - y_a)(y_2 - y_1) + (z_b - z_a)(z_2 - z_1)$$

$$b_s = (x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2$$

$$c_s = (x_b - x_a)^2 + (y_b - y_a)^2 + (z_b - z_a)^2$$

$$d_s = - a_s$$

$$e_s = (x_1 - x_a)(x_2 - x_1) + (y_1 - y_a)(y_2 - y_1) + (z_1 - z_a)(z_2 - z_1)$$

$$f_s = (x_1 - x_a)(x_b - x_a) + (y_1 - y_a)(y_b - y_a) + (z_1 - z_a)(z_b - z_a)$$

If $\lambda < 0$ or $\lambda > 1$ then the influence will originate outside the segment $P_a P_b$, so g will be null for this case. If $0 < \lambda < 1$, we have three possibilities; $\mu < 0$, meaning P_1 is closest to the cylinder, in this case

we take $\mu=0$. If $0<\mu<1$ then the segment P_1P_2 is closest to the cylinder. And if $\mu>1$, then P_2 is closest to the cylinder and we take $\mu=1$. So when $0<\lambda<1$, we know $P_{ax}(\lambda)$ and $P_p(\mu)$ and we can find the direction and magnitude of the vector $(P_{ax}-P_p)$, which will be related to g in that it has the same direction and its magnitude minus the radius of the cylinder is the distance d which we use for evaluate the magnitude of g using the decreasing function previously defined.

General Obstacle

In order to use the three elements defined previously, let us consider a tetrahedron as an example. Let the tetrahedron be defined by its four vertices P_a, P_b, P_c and P_d. In order to find the influence created by the tetrahedron, we find the greatest of the influences created by the following elements: triangles $(P_aP_bP_c)$, $(P_aP_cP_d)$, $(P_aP_dP_b)$, $(P_cP_bP_d)$, cylinders (R=0) (P_aP_b), (P_aP_c), (P_aP_d), (P_bP_c), (P_bP_d), (P_cP_d) and spheres (R=0) (P_a), (P_b), (P_c), (P_d). This procedure can be extended any obstacle of arbitrary shape by approximating the obstacle by a group of triangles like which is very similar to the method of triangulation used by topographers and surveyors.

VITA

Luis Angel Loeff

Candidate for the Degree of

Master of Science

Thesis: AN ALGORITHM FOR COMPUTER GUIDANCE OF A MANIPULATOR IN BETWEEN OBSTACLES

Major Field: Mechanical Engineering

Biographical:

Personal Data: Born in Montevideo, Uruguay, September 6, 1945, the son of Jose Loeff and Eta Borow de Loeff.

Education: Graduated from Zorrilla High School, Montevideo, Uruguay, in December, 1960; graduated from Engineering Preparatory IAVA School, Montevideo, Uruguay, in May, 1964; graduated as Ingeniero Industrial in Facultad de Ingeniería, Universidad de la Republica, Montevideo, Uruguay, in March, 1971; completed requirements for the Master of Science degree at Oklahoma State University in December, 1973.