RSD-TR-11-87

A Variational Dynamic Programming Approach to Robot Path Planning With a Distance-Safety Criterion

S.H. Suh

K.G. Shin

Department of Electrical Engineering and Computer Science
The University of Michigan
Ann Arbor, Michigan 48109

August 1987

Center for Research on Integrated Manufacturing

Robot Systems Division
College of Engineering
The University of Michigan
Ann Arbor, Michigan 48109-2110

TABLE OF CONTENTS

1. Introduction 2

2. Cost Function 5

3. Variational Dynamic Path Equations 7

4. Derivation of Channels 15

5. Conclusion 22

Acknowledgement 23

References 23

Appendix A 24

Appendix B 25

Figures 26

A Variational Dynamic Programming Approach to Robot Path Planning With a Distance-Safety Criterion[1]

Suk-Hwan Suh and Kang G. Shin

Robot Systems Division
Center for Research and Integrated Manufacturing
The University of Michigan
Ann Arbor, MI 48109–2122

August 26, 1987

Abstract

A new approach to robot path planning is developed by considering both the traveling distance and the safety of a robot. Incorporation of robot safety into path planning is important not only because of the uncertainties in the robot dynamics during path execution, but also because of the inaccuracies in the geometric modeling of obstacles. A computationally efficient algorithm is developed to find a near-optimal path with a weighted distance-safety criterion by using a variational calculus and dynamic programming (VCDP) method.

The algorithm is readily applicable to any factory environment by representing the free workspace as channels. A method for determining these channels is also proposed. Although it is developed for 2D problems, our method can be easily extended to a class of 3D problems. Numerical examples are presented to demonstrate the utility and power of this method.

[1]This work was supported in part by the NASA Johnson Space Center under grant NCC–9–16, the NSF under grant ECS–8409938, and the Industrial Affiliates Program, Robot Systems Division, Center for Research on Integrated Manufacturing (CRIM), The University of Michigan. All correspondence regarding this paper should be addressed to Kang G. Shin.

1 Introduction

Optimal (in some sense) control of industrial robots is of practical importance to various robot-based automation systems. Such a control is usually achieved through a two stage optimization: off-line path planning followed by on-line path execution. Depending on the system objective, both the path planning and path tracking problems have to be solved by optimizing suitable criteria subject to some constraints. This paper deals with the first stage, presenting a near-optimal solution to the robot path planning problem with a weighted distance-safety criterion.

The traveling distance has been the primary object to minimize in most conventional robot path planning approaches because the shortest distance path may reduce the robot's traveling time and the required mathematical complexity. However, another factor which should not be ignored during robot path planning is robot safety during path execution. Robot safety becomes important, especially when there are non-negligible uncertainties in both the robot dynamics during path execution and the environmental information such as obstacles. Thus, the simultaneous consideration of distance and safety needs to be called for during robot path planning.

The issue to be addressed in this paper can be made clear by comparing two well-known paths: the *shortest path* (SP) and the *center-line path* (CLP). A SP can be found by using a visibility graph in 2D [7] and is attractive for many cases. However, a SP may require the robot to travel too close to obstacles (actually, the SP touches them), and hence possesses a high risk of colliding with obstacles (Fig. 1). Thus, it may not even be desirable at all when robot safety is a major concern.

The safety of a path can be quantified by the clearance between the path and obstacles. Naturally, the larger its clearance, the safer a path will be. If robot safety is the only concern, one would choose a path providing the maximum clearance from obstacles. This path would traverse along the center-line of free space, i.e., the CLP. See Fig. 1 for an example CLP. As illustrated in this figure, a CLP could be considerably longer than a SP, indicating that the CLP is not desirable at all if the robot's traversal distance is a major consideration in path planning.

Following the above arguments, SP and CLP can be viewed as two opposite extremes with respect to the distance and safety criteria, respectively; neither of the two *alone* may be acceptable for the general case. It is thus necessary to develop a new path planning algorithm by striking a compromise between distance and safety.

The safety of a path has *not* been considered *explicitly* in almost all known path planning approaches except for the one in [7]. The path safety in [7] was obtained by first growing each obstacle by a specified amount (i.e., the margin of safety) and then applying the visibility graph method. A visibility graph is based on the necessary condition that a SP is made up of line segments. Though the method of growing obstacles and then using a visibility graph is simple and attractive in many cases, a potential problem with this method is that the only feasible path could have been eliminated as a result of growing obstacles [8]. Moreover, it may be very difficult to determine the degree of enlargement of obstacles during path planning because of its dependence on the utilization of the workspace as well as the uncertainties in the robot dynamics during path execution. Moreover, since a visibility graph can be set up only when the starting and ending points are given, a new visibility graph has to be constructed every time the starting and ending points are changed.

Other path planning approaches in the literature are based on the decomposition of free space into *geometric primitives*, such as cones and cylinders. Brooks [2] used overlapping generalized cones to represent the free space. Generalized cones are formed to allow translations and rotations. A robot path traverses along the cone axes and avoids obstacles generously, i.e., an implicit consideration of robot safety. One problem with this is that it is awkward to represent a large free space with generalized cones. As a remedy for this problem, Kuan *et. al* [5] proposed a hybrid representation of free space by using non-overlapping convex polygons for large free spaces and generalized cones for narrow regions. Singh *et. al* [10] decomposed a free space into several rectangular areas, and Chatila [3] used convex polygons to represent the free space.

These decomposition methods can be used to form a connectivity graph in which a node represents a geometric primitive and an arc describes the neighboring relationship between primitives. A graph search technique, such as the A^* algorithm, is then used to find the minimum (or near-minimum) cost path. The cost between a pair of nodes is the distance of the straight-line segment connecting the two points, each of which lies in the geometric primitive represented by each of the two nodes. The use of a straight-line segment as a *path primitive* for connecting two geometric primitives can be justified when path length is used as the cost function, although it may be necessary to remove the sharp corners on the resulting piecewise straight-line path. However, if the cost function contains more than just path length, use of a line segment as the path primitive is difficult to justify, particularly when the geometric primitive used covers a relatively large free

space.

To alleviate the above problem, one may reduce the coverage of a geometric primitive by placing a finer grid on the free space, which will, unfortunately, result in an excessive amount of computation. For example, the method in [11] placed a grid on the free space and formed a search graph by representing a grid point as a node with arcs to its eight neighboring nodes. Then, graph search is performed by using the A^* algorithm with a cost function that keeps the path from getting too close to obstacles. As the grid gets finer, the required computation will become excessive even if heuristics were used. Furthermore, the path connecting the grid points with straight-line segments could contain as many corner points as the number of grid points that the path goes through.

One way to overcome the above computational problem without sacrificing the accuracy of the solution is to use a path primitive that minimizes the cost function on a relatively *coarse* grid. To reduce the computational requirement, the grid size should be chosen as large as possible within the limit that the use of the path primitive can be justified. In such a case, the selection of a path primitive itself will become an optimization problem. Using a coarse grid and solving the optimization problem to connect the grid points, we develop in this paper a new approach to robot path planning that is accurate and computationally efficient.

The path planning approach presented here is largely divided into two parts. The first part considers the problem of finding an optimal path for a bounded free space called a *channel*, and derives the solution paths using a variational calculus and dynamic programming (VCDP) technique. The second part deals with the problem of deriving the channels from the general workspace filled with obstacle polygons. Specifically, the paper is organized as follows. In Section 2 we discuss the cost function which includes both traveling distance and robot safety. In Section 3, the path planning problem is solved using VCDP. Issues of the computational complexity and a 3D extension are also discussed. Section 4 describes how to determine channels for a given workspace. The paper concludes with a few remarks in Section 5.

Throughout the paper, a robot is assumed to be a disk in 2D or a sphere in 3D, thus allowing the robot to be treated as a point by growing obstacles as large as the radius of the disk or sphere.

2 Cost Function

Our path planning approach begins with the selection of the cost function to be used to measure the goodness of a path. Let a path P be described by a parameterized curve connecting the starting and ending points, denoted by $\{x(\omega),\ \omega_0 \leq \omega \leq \omega_f\}$, where ω is the parameter describing the curve [9],[2] and ω_0 and ω_f are respectively initial and final values of the parameter. Further, let $\mathcal{O}$ be the space occupied by obstacles, O_i, $i = 1, \ldots, n_{\mathcal{O}}\}$.

Denote the cost function for a path $P = \{x(\omega),\ \omega_0 \leq \omega \leq \omega_f\}$ by

$$C(P) = \int_{\omega_0}^{\omega_f} L(x(\omega), \omega, \mathcal{O})\ d\omega. \tag{1}$$

Then, we want to minimize (1) subject to the obstacle avoidance constraint

$$\{x(\omega),\ \omega \in [\omega_0, \omega_f]\} \cap \mathcal{O} = \phi, \tag{2}$$

and the boundary conditions

$$x(\omega_0) = S,\ \ x(\omega_f) = E, \tag{3}$$

where S and E are the starting and ending points.

The form of the cost function characterizes the type of path planning. Considering both distance and safety, the cost function is represented by:

$$C(P) = \mathcal{D}(P) + \lambda \mathcal{S}(P),\ \ \lambda \geq 0, \tag{4}$$

where $\mathcal{D}(P)$ and $\mathcal{S}(P)$ represent costs associated the length and safety of P, respectively, and λ is the relative weighting between the two.

The distance cost of a path is defined as the arc length of the path. The safety cost of a path can be defined in terms of the deviation of a path from the safest path or the center-line path (CLP). Note that an exact CLP for a given workspace containing polygon obstacles can be derived by a generalized Voronoi diagram [6], and generalized Voronoi edges are made up of both straight-line and curve segments. An approximate CLP which is made up of straight-line segments only can also be found by decomposing the free space into convex polygons and connecting the bisection points of the common edges of polygons as will be discussed in Section 4.2. For clarity of presentation, it is assumed in this section that the CLP is given and made up of line segments.

[2]For example, the parameter may represent the arc length of the path.

Consider a CLP traversing a set of free convex polygons such that the starting and ending points are located, respectively, in the first and last free convex polygons. Suppose a path connecting the starting and ending points passes through these free convex polygons. Then, the path can be parameterized with respect to the CLP, [3] and the path's deviation from the CLP, called the *center-line deviation*, is defined as:

$$\int_{\omega_0}^{\omega_f} \| CLP(\omega) - x(\omega) \| \, d\omega, \tag{5}$$

where $\| \cdot \|$ represents the Euclidean norm and ω is the path parameter. Hence a safer path implies a smaller center-line deviation.

When the safety cost of a path is defined by Eq. (5), the following two aspects must be taken into consideration. First, since the center-line deviation may increase as path length increases, it should be normalized by, for example, the path length or the interval of the path parameter to avoid double counting the part of cost contributed by the path length or the interval of the path parameter. Second, since the CLP traverses the free spaces of varying clearance from obstacles, one unit center-line deviation at a free space would have a different degree of safety from that at another free space. For example, consider the two points P_1 and P_2 in Fig. 2, both of which deviate from the CLP by the same amount. Obviously, P_1 in Region 1 has a higher risk of collision than that of Region 2. Thus, it is necessary to scale the normalized center-line deviation on the basis of the *criticality* of a unit center-line deviation in each region. The scaling constant $\beta(\omega)$ should be inversely proportional to the clearance of each region.

Considering all the aspects mentioned above, the safety cost of a path is defined as follows.

$$S(P) = \int_{\omega_0}^{\omega_f} \gamma(\omega) \, \| CLP(\omega) - x(\omega) \| \, d\omega, \tag{6}$$

where $\gamma(\omega) = \beta(\omega)/(\omega_f - \omega_0)$. Note that this safety cost is obtained by normalizing the center-line deviation (5) with the interval of the path parameter, $\omega_f - \omega_0$, followed by multiplying the scaling constant, $\beta(\omega)$.

In a given workspace, there could be several approximate[4] CLPs connecting the starting (S) and ending (E) points. Let a *channel* be a bounded free space in which there exists at least one collision-free path between S and E. Then, there exists a CLP for every channel. For each channel, a minimum cost path, called a *local minimum cost path* (LMCP), may be found, and the *global minimum cost path* (GMCP) is the minimum

[3]The details of parameterization will be given in Section 3.

[4]since the starting and ending points may not be on the center-line paths.

among the LMCPs. A detailed solution approach to the LMCP problem is presented in Section 3. The derivation of the GMCP will be treated in Section 4.

3 Variational Dynamic Path Equations

In this section, the problem of finding an optimal path for a given channel is dealt with. First, we will consider a simple problem called the *Path Primitive Problem*, for which a solution will be derived. Then, we will treat a more complex and general problem, a solution algorithm for which will be developed. Later in this section, issues on the 3D extension of our algorithm are also discussed.

3.1 Path Primitive Problem

Suppose the robot is to move from a starting point S to an ending point E in a given channel described by the center-line segment, C, and two obstacle boundary lines, U and L, as shown in Fig. 3. We want to find a path connecting S and E that minimizes a weighted distance and safety cost while avoiding collision with obstacles.

Placing a rectangular coordinate frame with its origin at one end of C and the horizontal axis coinciding with C, a path can be defined as a function x of the path parameter $\omega \in [\omega_0, \omega_f]$, where $S = (\omega_0, x_0)$ and $E = (\omega_f, x_f)$. Let the equation of the boundary lines and C be given as:

$$L(\omega) = a_L\omega + b_L, \quad \omega_{L_L} \leq \omega \leq \omega_{L_U}, \tag{7}$$

$$U(\omega) = a_U\omega + b_U, \quad \omega_{U_L} \leq \omega \leq \omega_{U_U}, \tag{8}$$

$$C(\omega) = 0, \quad 0 \leq \omega \leq \omega_{C_U}, \tag{9}$$

where a_i, b_i for $i = L, U$ are coefficients of boundary line equations, $\omega_{L_U} - \omega_{L_L}$ and $\omega_{U_U} - \omega_{U_L}$ are the curvilinear lengths of projected L and U on the horizontal axis, and ω_{C_U} in Eq. (9) is the length of C.

To show how the scaling constant can be obtained, consider the following case which corresponds to *Case A* in Fig. 6(a)

$$\omega_0 \geq 0, \quad \omega_f \leq \omega_{C_U}. \tag{10}$$

Since the horizontal axis of the coordinate system coincides with the center-line segment, the center-line deviation of the path becomes $\int_{\omega_0}^{\omega_f} \| C(\omega) - x(\omega) \| \, d\omega = \int_{\omega_0}^{\omega_f} |x(\omega)| \, d\omega$.

Note that the path length can be computed as $\int_{\omega_0}^{\omega_f} \sqrt{1+\dot{x}(\omega)^2}\, d\omega$. The scaling constant for the channel is defined as a reciprocal of the average clearance, which can be obtained by computing the distances between the midpoint of C and the upper and lower boundary lines:

$$\overline{\beta} = \left[\|(\frac{1}{2}\omega_{C_U}, 0) - U\| + \|(\frac{1}{2}\omega_{C_U}, 0) - L\| \right]^{-1} \tag{11}$$

Note that $\overline{\beta}$ for different cases can be computed similarly.

Once the scaling constant is obtained, the cost function (4) becomes

$$\begin{aligned} C(S,E) &= \mathcal{D} + \lambda \mathcal{S} \\ &= \int_{\omega_0}^{\omega_f} \left[\sqrt{1+\dot{x}(\omega)^2} + \lambda\gamma\, |x(\omega)| \right] d\omega, \end{aligned} \tag{12}$$

where $\gamma = \overline{\beta}\,'(\omega_f - \omega_0)$, and the path primitive problem is to find x^* which minimizes (12) subject to the following constraints:

$$a_L\omega + b_L \le x(\omega) \le a_U\omega + b_U, \quad \forall\, \omega \in [\omega_0, \omega_f], \tag{13}$$

$$x(\omega_0) = x_0, \quad x(\omega_f) = x_f. \tag{14}$$

This is a fixed-terminal-point variational problem. It becomes an *unconstrained* problem since the constraint (13) can automatically be met as stated below.

Proposition 1: The constraint (13) is redundant, $\forall\, \lambda \in [0, \infty]$, if S and E are within the free space.

Proof: Suppose $x^*(\omega)$ sways above the upper-boundary line U. Then there exist two distinct points, one to break-out, $B_1 = (\omega_1, x_1)$, and the other to break-in, $B_2 = (\omega_2, x_2)$, to the line, since the path must be continuous and $x_0,\ x_f \le U(\omega),\ \forall\, \omega \in [\omega_0, \omega_f]$ (Fig. 4). For the interval $[\omega_1, \omega_2]$, clearly the straight-line path connecting B_1 and B_2 is superior to x^* in the sense of both distance and the center-line deviation. Thus, an optimal path cannot traverse over the upper boundary line. The same argument holds for the lower boundary line L. Hence, constraint (13) is met automatically. □

The solution to the unconstrained variational problem may be obtained by solving the Euler equation [4]:

$$0 = \frac{\partial g}{\partial x}(x^*(\omega), \dot{x}^*(\omega), \omega) - \frac{d}{d\omega}\left[\frac{\partial g}{\partial \dot{x}}(x^*(\omega), \dot{x}^*(\omega), \omega) \right], \tag{15}$$

or

$$k\ sig(x^*(\omega)) = \frac{\ddot{x}^*(\omega)}{(1+\dot{x}^{*2}(\omega))^{3/2}}, \tag{16}$$

where g is the integrand of (12), $k = \lambda\gamma$, and $sig(x) = 1$ if $x < 0$, -1 if $x < 0$, and undefined if $x = 0$. Notice that the Euler equation is a nonlinear second order differential equation, which cannot usually be solved analytically. Thus, one has to resort to numerical integration. However, in this case, a nonlinear, two-point boundary-value problem must be solved. The problem is in general very difficult because of both the split boundary values and the nonlinearity of the differential equation.

For computational tractability, we use a weighted squared sum of the distance and safety costs in place of the cost functional (12),

$$J(S,E) = \int_{\omega_0}^{\omega_f} \left[\left(1 + \dot{x}^2(\omega)\right) + \lambda\gamma\, x^2(\omega)\right] \, d\omega. \tag{17}$$

It is worth noting that the quadratic form is often employed in other optimal control problems for mathematical tractability, e.g., the LQ problem [1].

Proposition 2: The value of the cost functional J must be used only for finding an optimal path x^* for a given set of starting and ending points within each channel, not as a means for evaluating the performance of x^*. Suppose x_1^* and x_2^* are optimal paths connecting two different sets of starting and ending points, i.e., minimizing $J(S_1, E_1)$ and $J(S_2, E_2)$, respectively, where $(S_1, E_1) \neq (S_2, E_2)$. Then, $J(S_1, E_1) > J(S_2, E_2)$ does not necessarily mean that x_2^* is superior to x_1^* (see Appendix A for the proof of this proposition). For such a purpose, the original cost function, $C(S_i, E_i)$, must be used, not the weighted squared sum (17).

In what follows, a solution to the path primitive problem with the cost function (17) is derived. The Weierstrass-Erdmann corner conditions [4] assure that there is no corner in the optimal path (see Appendix B for a detailed description). The Euler equation (15) is now given as:

$$\ddot{x}^*(\omega) - \lambda\gamma x^*(\omega) = 0. \tag{18}$$

Since Eq. (18) is linear in x^* with constant coefficients, it can be readily solved to get:

$$x^*(\omega) = c_1 e^{\sqrt{k}\,\omega} + c_2 e^{-\sqrt{k}\,\omega}, \tag{19}$$

where c_1 and c_2 are the integration constants determined by the two boundary conditions (14), and $k = \lambda\gamma$.

Paths with the various values of λ for two sets of starting and ending points are plotted in Figs. 5(a)–(b). It is shown that a) all the paths are smooth as evidenced by the corner conditions, b) as the value of λ increases (decreases) the path tends to be closer to

the center-line (straight-line), and c) all the paths traverse inside the convex hull defined by the *four points*, $(\omega_0, x_0), (\omega_0, 0), (\omega_f, 0), (\omega_f, x_f)$.

In fact, b) and c) can be proved by finding an asymptotic curve as follows. As $\lambda \rightarrow 0$, the Euler equation (18) becomes $\ddot{x}^*(\omega) = 0$, i.e., $x^*(\omega) = c_3\omega + c_4, \ \forall\, \omega \in [\omega_0, \omega_f]$, where c_3 and c_4 are determined by the two boundary conditions (14). As $\lambda \rightarrow \infty$, the first term in (18) becomes negligible, implying that $x^*(\omega) \rightarrow 0$. Due to the boundary conditions (14), the asymptotic curve becomes: $x^*(\omega) = \begin{cases} x_0 & \text{if } \omega = \omega_0 \\ 0 & \text{if } \omega_0 < \omega < \omega_f \\ x_f & \text{if } \omega = \omega_f \end{cases}$, indicating that the asymptote in this case is straight-line segments connecting *four points* in sequence: $(\omega_0, x_0), (\omega_0, 0), (\omega_f, 0)$, and (ω_f, x_f).

Since the same reasoning as that of Proposition 1 holds for the weighted squared distance and safety cost, the obstacle boundary constraint (13) is automatically met as far as the calculation of x^* by minimizing (17) is concerned. The following proposition generalizes the condition that the obstacle boundary constraint can be ignored.

Proposition 3: The obstacle-avoidance constraint is redundant $\forall\, \lambda \geq 0$, if $S, E \in F$, where F is a convex set representing the free space.

Proof: Regardless the value of λ, the optimal path traverses inside the convex hull defined by the four points. Since the center-line segment traverses inside the free space, the four points, $(\omega_0, x_0), (\omega_0, 0), (\omega_f, 0), (\omega_f, x_f) \in F$, if $S, E \in F$. Thus, the convex hull $\subset F$, and $\{x^*(\omega),\ \omega_0 \leq \omega \leq \omega_f\} \in F$. □

The use of the x^* given in Eq. (19) can be generalized by the following proposition.

Proposition 4: Even if the condition (10) is not met, the x^* given in Eq. (19) is still an optimal path.

Proof : Let *Case A* : $\omega_0 \geq 0,\ \omega_f \leq \omega_{C_U}$, *Case B* : $\omega_0 \geq 0,\ \omega_f \geq \omega_{C_U}$, *Case C* : $\omega_0 \leq 0,\ \omega_f \leq \omega_{C_U}$, and *Case D* : $\omega_0 \leq 0,\ \omega_f \geq \omega_{C_U}$.

To derive the solution for Case B (Fig. 6b), the functional (17) is represented by:

$$J = \int_{\omega_0}^{\omega_f} \left[(1 + \dot{x}^2(\omega)) + \lambda\, \gamma \parallel (\omega, x(\omega)) - C \parallel^2 \right] d\omega. \tag{20}$$

Note that the center-line deviation in (20) is represented in terms of the distance between the point coordinate of $(\omega, x(\omega))$ and the center-line segment C. Also, the representation inside the norm is the same as $x^2(\omega)$ in (17) when $\omega_0 \geq 0,\ \omega_f \leq \omega_{C_U}$. The center-line

deviation is

$$\begin{aligned}\int_{\omega_0}^{\omega_f} \|(\omega, x(\omega)) - C\|^2 &= \int_{\omega_0}^{\omega_{C_U}} \|(\omega, x(\omega)) - C\|^2 \, d\omega + \int_{\omega_{C_U}}^{\omega_f} \|(\omega, x(\omega)) - C\|^2 \, d\omega \\ &= \int_{\omega_0}^{\omega_{C_U}} x^2(\omega) \, d\omega + \int_{\omega_{C_U}}^{\omega_f} [\, x^2(\omega) + (\omega - \omega_{C_U})^2 \,] \, d\omega \\ &= \int_{\omega_0}^{\omega_f} x^2(\omega) \, d\omega + \int_{\omega_{C_U}}^{\omega_f} (\omega - \omega_{C_U})^2 \, d\omega. \end{aligned} \tag{21}$$

Since for given ω_{C_U} and ω_f the second term in (21) is a constant, indicating that the $x^*(\omega)$ given in Eq. (19) is also the solution for *Case B*. Similarly, the same conclusion can be drawn for Cases C and D. □

As a result of Propositions 3 and 4, $\{ x^*(\omega), \omega_0 \leq \omega \leq \omega_f \}$ given in Eq. (19) can be used as an optimal path primitive for any pair of (S, E) in a convex channel characterized by a single center-line segment and two obstacle boundary lines. This includes the case when the channel is given as a triangle where one of the obstacle boundary lines is a point.

In the next subsection, we will discuss a solution method for a more complex case where the channel is characterized by two or more center-line segments.

3.2 Multi-Segmented Channel Problem

Consider a free space where the CLP is composed of two line segments as shown in Fig. 7. Place a dividing line (called a *barricading line* or simply a *barricade*) passing through the CLP's corner point such that the CLP in each region is a straight-line segment. Then, a local coordinate frame is placed at the beginning of each center-line segment in the same fashion as that of Path Primitive Problem (Fig. 7).

Since all paths must cross over each barricading line, it is necessary to find the "optimal" crossing point, where the optimal path intersects the barricade. To determine the optimal crossing point, a barricading line is discretized into a finite number of equally spaced points called *gates*. Let $G_i(j)$ and M be, respectively, the j^{th} gate of barricade i and the number of gates.[5] Then the location of $G_i(j)$ is represented by

$$G_i(j) = G_i(1) + (j - 1)\eta_i \, \vec{\imath}, \tag{22}$$

[5]We assumed the same number of gates for all barricades for notational simplicity. It is trivial to relax this assumption.

where $\vec{\imath}$ is a unit vector from one end, $G_i(1)$, to the other end, $G_i(M)$, and $\eta_i = \ell_i/(M-1)$ and ℓ_i is the length of barricade i.

The optimal path passing through $G_i(j)$ can be found by applying Eq. (19) twice with starting and ending points of $(S, G_i(j))$ and $(G_i(j), E)$, and the optimal path connecting (S, E) is the one passing through the gate that minimizes the cost

$$C(S,E) = \min_j \left[C_1(S, G_i(j)) + C_2(G_i(j), E)\right], \tag{23}$$

where $C_k(P,Q)$, $k = 1,2$ is the cost of path connecting P and Q in the k^{th} region, and is computed by Eq. (12).[6] Note that due to Proposition 2, Eq. (12) instead of Eq. (17) must be used for computing the cost of a path.

The method used for a two-segmented channel can be applied to a multi-segmented channel. Suppose a free space is characterized by N center-line segments and $N-1$ barricades (Fig. 8). For notational convenience, the starting and ending points are treated as barricades with $M = 1$. These are not real but *pseudo* barricades. Thus, there is a total of $N+1$ barricades, where $G_1(1)$ and $G_{N+1}(1)$ are the starting and ending points.

Let $I = \{g_1, \ldots, g_{N+1}\}$ be the set of gates that a path traverses, where g_i is the gate number in barricade i. Then, the multi-segmented channel problem is to find a gate set whose corresponding path has the minimum cost. The cost for the gate set I is given by

$$C(I) = \sum_{i=1}^{N} C_i(G_i(j), G_{i+1}(k)), \quad \text{for some } j,k \in I \tag{24}$$

where $C_i(G_i(j), G_{i+1}(k))$ is the cost of x_i^- connecting the gates $G_i(j)$ and $G_{i+1}(k)$.

The optimal gate set is determined by the dynamic programming (DP). The optimization procedure starts at barricade N. (Recall that barricade $N+1$ is the ending point.) The cost of a path passing through gate j on barricade N, denoted by $C_N(j)$, can be computed once $x_N^-(j)$ is determined using $G_N(j)$ and $G_{N+1}(1)$ as the starting and ending points, respectively, i.e., $C_N(j) = C_N(G_N(j), G_{N+1}(1))$, and the next gate pointer at gate j on barricade N, $d_N(j) = 1$, $\forall\, j = 1, \ldots, M$.

The cost at gate j on barricade $N-1$ becomes

$$C_{N-1}(j) = \min_k \left[\, C_{N-1}(G_{N-1}(j), G_N(k)) + C_N(k) \,\right], \quad k = 1, \ldots, M, \tag{25}$$

[6]Precisely speaking, Eq. (12) is the cost function of Case A in Proposition 4, and a general cost function covering the other cases needs to replace the term $|x(\omega)|$ by $\|(\omega, x(\omega)) - C\|$ in Eq. (12). For convenience, Eq. (12) will henceforth be referred to as a cost function for all the cases.

and the pointer $d_{N-1}(j)$ will be the k that minimizes the magnitude of $[\cdot]$ in Eq. (25). Recursive equations can be obtained by replacing N and $N-1$ in the above with $i+1$ and i, respectively.

The DP algorithm is summarized as follows:

- *Step 1*: Initialization

  1. Read in center-line segments and barricades data.
  2. Form gates: $G_i(j)$, $i = 1, \ldots, N+1$; $j = 1, \ldots, M$.
  3. Set $i := N$.
  4. For $j = 1, \ldots, M$,[7] set $d_i(j) := 1$, and compute $C_i(j) = C_i(G_i(j), E)$.

- *Step 2*: Termination check
  Set $i := i - 1$.
  If $i = 0$, stop. Otherwise go to *Step 3*.

- *Step 3*: Continuation
  For $j = 1, \ldots, M$,

  1. Set $C_i(j) := \infty$.
  2. For $k = 1, \ldots, M$,
     Compute $X = C_i(G_i(j), G_{i+1}(k)) + C_{i+1}(j)$
     If $X < C_i(j)$, then $C_i(j) := X$ and $d_i(j) := k$.

  Go to *Step 2*.

The computational complexity of the DP algorithm is as follows. For each region bounded by two barricades, there are M^2 pairs of gates except for the first and last regions in each of which there are M pairs of gates. Thus, there are $2M + (N-2)M^2$ pairs of gates for which the cost function (12) is evaluated. Note, however, that since the path equation is given in closed form, the computational cost is very low.

Since the path equation is given with reference to a local coordinate frame whose origin is at one end of the center-line segment, a coordinate transform may be necessary

[7] If $N = 1$, let $M = 1$.

to convert the path in the local coordinate frame into the one in the world coordinate frame, W. For this purpose, the following transform equation can be used:

$$\begin{bmatrix} {}^{W}\omega \\ {}^{W}x \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} \omega \\ x \end{bmatrix} + \begin{bmatrix} \omega^{S} \\ x^{S} \end{bmatrix}$$
$$= R(\theta) \begin{bmatrix} \omega \\ x \end{bmatrix} + \begin{bmatrix} \omega^{S} \\ x^{S} \end{bmatrix}, \tag{26}$$

where R is the transform matrix representing the rotation of the local coordinate frame relative to the world coordinate frame, and θ is the rotational angle of the local coordinate frame defined as the counter-clockwise angle between the center-line segment, C, and the horizontal axis of the world coordinate frame, and (ω^S, x^S) represents the translation of C from the origin of the world frame.

An example of the behavior of optimal paths with respect to a rectangular obstacle while varying the weighing factor, λ, is shown in Fig. 9. There is a three-segmented channel in this example, which was simulated with 20 gates on each of the two barricading lines. It is shown that an optimal path tends to traverse away from (close to) the obstacle as the value of λ increases (decreases). The paths with λ close to the extreme values have visual corners, since in such cases the paths are approaching the asymptotes, i.e., SP or CLP. However, they are relatively smooth for mid-range values of λ. Observe that regardless of the value of λ, there do not exist ridges around the gates where the two path primitives are joined, since such a path cannot be optimal.

3.3 3D Extension

We have presented the VCDP algorithm to determine an optimal path in 2D. In 2D, the primitive of the safest path is a straight-line, i.e., the CLP. In case of 3D, however, the CLP is not a line but a plane in general. Therefore, the VCDP algorithm is difficult to apply to general 3D problems. However, if 3D problems are restricted to the safest paths that are represented by line segments only, the VCDP algorithm can be used for them with a minor modification.

Consider a room filled with polyhedral objects. Suppose an *articulated cylinder* is placed as shown in Fig. 10. Regard the axis of the cylinder as a CLP and its cross-sectional diameter as the clearance. Then, the cross-sectional circle where the CLP is bent becomes a barricade. The articulated cylinder may be thought of as a channel. Note, however, that the approximation of the free space with an articulated cylinder wastes the

free space. To minimize the wasted volume, the articulated cylinder should be placed such that the largest diameter can be obtained.

With the above arrangement in 3D, the VCDP algorithm can be directly applied to 3D problems. The differences are: a) gates are generated by discretizing the barricading plane (Fig. 11), and b) the path equation is derived such that the following functional is minimized:

$$J = \int_{\omega_0}^{\omega_f} \left[(1 + \dot{x}(\omega)^2 + \dot{y}(\omega)^2) + \lambda\gamma(x(\omega)^2 + y(\omega)^2) \right] \, d\omega, \tag{27}$$

where $\gamma = \overline{\beta}/(\omega_f - \omega_0)$, and $\overline{\beta}$ is the reciprocal of the average diameter of each cylinder. Details of the solution to this 3D problem are omitted since it is very similar to the 2D case.

4 Derivation of Channels

In the previous section, the VCDP algorithm was developed to determine the optimal path equations for a *given* channel characterized by multiple center-line segments. For the development of this algorithm, the channels and their parameters (i.e., CLP and barricades) are assumed to have been given. In this section we shall develop a method for deriving channels and their parameters for the workspace.

4.1 Describing Free Space as Channels

To apply the VCDP algorithm, it is necessary to describe the free space as channels, each of which is defined as a closed free space that can be accessed only through its two entrances. Boundaries of a channel are two piecewise straight lines connecting its two entrance edges. It is important to note that the free space inside the boundaries of a channel must be completely free of islands of forbidden area (generated by obstacles). In other words, *holeless* channels are to be placed in the *punched-hole* shape of the free space. Thus, to describe an open free space (the complement of the areas occupied by obstacles) in terms of channels, the workspace is divided into two areas (Fig. 12): the *obstacle-cluster area* (OCA) which is the convex hull of obstacle polygons, and the *corridor area* which is the complement of the OCA. Then, the free space in the corridor area can be described by a single channel whose boundaries are defined by the OCA and the workspace itself.

In contrast with the corridor area, it is usually difficult to describe the free space in OCA with a single channel. Since those edges of the obstacle-cluster convex hull which are not the edges of obstacle polygons[8] can be considered as the entrances to OCA, the free space in OCA is represented by several channels, one for each pair of entrances to the area.

For a given pair of entrances, channel boundaries are determined in two stages. The first stage is to identify polygons which define the shape of the channel. This is necessary since there may exist polygons which play no or a minor role in determining the shape of the "main road" of the free space connecting the channel's two entrances. Then, the channel boundaries are determined in the second stage by using the bounding polygons obtained in the first stage. The details of the two stages are given in the following subsection. For the simplicity of presentation, we assume that two entrances are at top and bottom, and that the channel boundaries to be determined are at left and right.[9]

4.1.1 Identification of Bounding Obstacles

Bounding polygons of a channel must be identified such that the free space within the channel is maximized. Such bounding polygons are iteratively obtained, each time by drawing a convex hull with previously identified bounding polygons and checking the relationship between the edges of the convex hull and obstacle polygons to be identified. The set of polygons defining the two entrances of a channel, called *entrance-defining polygons* (EDPs), is used as the initial set of bounding polygons. In the discussion that will follow, it is assumed that the OCA contains polygons other than EDPs. If it contains only EDPs, then the identification process will terminate at this point.

Suppose the two entrances of the convex hull, $CH(EDPs)$, are located at top and bottom, and the two non-entrance edges are located at left and right, i.e., "left" and "right" edges (Fig. 13a). Note that there exists only one edge if an obstacle polygon is used for defining both the top and bottom entrances (Fig. 13b). The initial bounding polygon set, $\mathcal{B}_1 = \{EDP\}$, can be decomposed into $\mathcal{B}_1^L$ and $\mathcal{B}_1^R$, where $\mathcal{B}_1^L$ ($\mathcal{B}_1^R$) is the set of left-(right-) bounding polygons that define the left (right) edge, L_1 (R_1) of $CH(\mathcal{B}_1)$.

[8]Specifically, an entrance is defined as an edge whose two endpoints do not belong to the same OCA-defining polygon. Note that the above definition is valid for the OCA consisting of both convex and concave obstacle polygons.

[9]Distinction between "top", "bottom", "left" and "right" edges is immaterial but used for the convenience of presentation.

Now, it is necessary to identify and then label three different types of obstacle polygons: left-bounding polygons, right-bounding polygons, and island-bounding polygons. Since the method of identifying right-bounding polygons is symmetric to that of left-bounding polygons, it is sufficient to describe those steps necessary to identify left-bounding and island-bounding polygons.

Identification of left-bounding polygons: Let EDP^L and EDP^R be those polygons defining L_1 and R_1, respectively. Then, the initial set of left-(right-) bounding polygons are $\mathcal{B}_1^L = \{EDP^L\}$ and $\mathcal{B}_1^R = \{EDP^R\}$. The set of left-bounding polygons is then expanded by adding those unlabeled obstacle polygons (excluding EDPs) that satisfy the conditions described below. When $i = 1$, an unlabeled obstacle polygon, O_j, is labeled "L" ("R" in the steps of identifying right-bounding polygons) if $O_j \cap L_i \neq \phi$. This is to find those obstacle polygons that intersect L_i between the two obstacle polygons defining L_i. When $i > 1$, the following condition is necessary to label O_j:

$$O_j \cap \{CH(\mathcal{B}_i) - CH(\mathcal{B}_{i-1})\} \neq \phi. \tag{28}$$

This additional condition is needed to identify obstacle polygons lying between L_i and L_{i-1}. Note that it is necessary to differentiate the labeling conditions for the cases of $i = 1$ and $i > 1$, since in the former case the additional condition (28) is used to identify the island of forbidden area as will be seen later in this subsection.

By applying the above conditions for every unlabeled polygon, a new expanded set of left-bounding polygons is formed. Termination conditions are: a) $\mathcal{B}_i^L \neq \mathcal{B}_{i-1}^L$, i.e., there is at least one obstacle polygon labeled in the i^{th} iteration, and b) $L_i \subset CH(\text{OCA})$, i.e., no more obstacle polygons left with which the current convex hull, $CH(\mathcal{B}_i)$, can be enlarged. If any of the termination conditions is satisfied, then go to the steps of identifying right-bounding polygons. Otherwise, let $i := i + 1$, and form a new expanded set of left-bounding polygons, $\mathcal{B}_i^L$, consisting of $\mathcal{B}_{i-1}^L$ and unlabeled OCA-defining polygon(s)[10] each located left to the top/bottom-leftmost OCA-defining polygons in $\mathcal{B}_{i-1}$. Then, repeat the steps of identifying left-bounding polygons. This process can be thought of as an attempt of "maximally expanding" the left-bounding polygon set.

Identification of island-bounding polygons: After identifying left- and right- bounding polygons, the polygons inside the original convex hull, $CH(EDPs)$, must be identified. Note that this process does not deal with those obstacle polygons that reside inside

[10]Two polygons if exist or one if not.

the convex hull and do not intersect all non-entrance edges, i.e., L_1 and R_1. Similarly to the above, island-bounding polygons are identified by the convex hull operation. The first step is to merge more than one polygon (if exist) into one polygon (an "island" of forbidden area) by the convex hull operation. The merged polygon is then treated as a single obstacle polygon and is labeled "RL" (Fig. 13c). Thus, in such a case, some free space (the shaded region in Fig. 13c) between obstacle polygons in the island is wasted. Note, however, that the merging operation is applied not to create unnecessarily many narrow channels which may exist otherwise. In other words, by the merging operation, the maximum number of channels to be defined in the free space for any pair of entrances is two as will be described in the following subsection.

4.1.2 Determination of Channel Boundaries

Channel boundaries are determined by the entrance-defining polygons and those labeled "L", "R", and "RL". Depending on the presence of island of forbidden area, two possible cases need to be considered. First, consider the case when there is no island of forbidden area. In this case, a single channel is defined as follows. Let $CH(L, R, EDPs)$ and $\mathcal{O}(L, R, EDPs)$ be, respectively, the convex hull and the occupied space of the obstacle polygons labeled "L", "R", and the entrance-defining polygons. Then, the configuration of channel can be represented as the convex hull complement of the forbidden area, i.e., $CH(L, R, EDPs) - \mathcal{O}(L, R, EDPs)$. Thus, the left (right) channel boundary in this case is obtained by connecting the vertex corresponding to the left (right) end of the top entrance edge to that of the left (right) end of the bottom entrance edge. Figs. 14(a)–(d) show three channels for an OCA consisting of three polygons, and one channel for the corridor area in Fig. 12. Note that the corridor area can be viewed as a special type of channel whose two entrances are identical.

In the case when there exists an island of forbidden area, two channels — left and right channels — are defined from the configuration of the free space, $CH(L, R, EDPs) - \mathcal{O}(L, R, RL, EDPs)$. The left (right) channel configuration is defined with obstacle polygons labeled L, RL, and EDPs (RL, R, and EDPs). In what follows, we describe a method to determine the configuration of the left channel only, since a) channel boundaries are determined by connecting the vertices along the contour of the configuration as described above, and b) a method symmetric to that of the left channel is applied to the right channel.

An exact configuration of the free space to be covered by the left channel is $CH(L, RL, EDPs) - O(L, RL, EDPs)$. Since an island of forbidden area labeled RL is always located inside the convex hull, the free space between RL and the right edge of the convex hull $CH(L, RL, EDPs)$ must be removed so that the right channel boundary can be determined. The removal is done by placing two bounding edges, each connecting two vertices — one from RL and the other from the top-right EDP or bottom-right EDP — such that the removed space is minimal. Figs. 15(a)–(b) show the boundaries of the left and right channels derived from the layout in Fig. 13(c).

4.2 Determining Channel Parameters

The channel boundaries obtained above may zigzag, and the free space within these boundaries may contain many "outlying" regions of the free space as shown in Figs. 14(a)–(c) and 15(a)–(b). These regions can be removed when the channel parameters, the CLP and barricades, are determined. Before discussing how to determine the channel parameters, it is worth noting that (i) the CLP should be determined to be the safest path in every free space because the VCDP algorithm is developed based on such an assumption, (ii) the free space between two successive barricades must be convex (due to Proposition 3), and (iii) the least number of barricades is desired for computational reasons.[11]

There is a trade-off between (i) and (iii), since the CLP's accuracy increases with the number of barricades used. For instance, the CLP connecting the bisection points of seven barricades in Fig. 16(a) is considered to be more accurate and, thus, safer than that of two barricades in Fig. 16(b). In general, as the number of edges of each basic convex polygon (in the free space) increases, the number of barricades decreases at the cost of the CLP's accuracy. We deal with this aspect by decomposing the free space into convex quadrilaterals and triangles, and keeping the number of quadrilaterals maximum.

The channel parameters are determined by decomposing the free space into non-overlapping convex polygons followed by labeling the edges of the decomposed polygons. This decomposition is straight-forward: choose the maximum concave angle in the free space bounded by all the edges, and try to include as many (up to four) contiguous vertices as possible to form a convex polygon in the free space. Thus, the basic convex polygon is either a triangle or a quadrilateral. The decomposition process will continue

[11]Note that the amount of computation required for the VCDP algorithm is approximately proportional to the number of barricades as mentioned in Section 3.2.

until the entire free space is divided into triangles and quadrilaterals only.

Then, the labeling process begins. Initially, all the polygon edges on channel boundaries are labeled "W" (indicating a wall) and the two entrance edges are labeled "B" (indicating barricades). Labeling is done on those polygons in which only one edge is not labeled. The unlabeled edge in a basic convex polygon is labeled "W", if all its other edges are labeled "W" (the polygonal free space is thus removed since it is an outlying region). Otherwise, it is labeled "B". The labeling process continues until all the edges are labeled. The edges labeled "B" then become barricades, and the straight-line segment connecting the bisection points of barricades becomes the CLP. The same method is applied to the channels representing the corridor area. In such a case, there is only one entrance edge which can be formed by taking a vertex from the workspace boundaries and connecting it to the nearest vertex in the obstacle-cluster convex hull. Fig. 17 illustrates the above method by using the same example in Fig. 14.

4.3 Setting Up a Graph

Since the CLP and the configuration of (the "main road" of) a channel are completely specified by barricades, it is necessary to store only the barricade edges to describe a channel. Since there are $\binom{n_e}{2}$ pairs of entrances, where n_e is the number of entrances in OCA[12], and a maximum of two channels are defined for each pair, the entire workspace including the corridor area can be described by at most $2\binom{n_e}{2} + 1$ sets of barricades.

To describe the workspace systematically, a graph is set up where each node represents the free convex polygon in the corridor area or the free space in OCA. Two nodes are joined by an arc if the nodes share a common barricade. Fig. 18b shows such a graph obtained from the layout of Fig. 12. Note that the arcs connected to a node of OCA are entrances to the free space represented by the node, and supplementary information between these arcs is given in the form of the ordered set of barricades as shown in Fig. 18c. Note that the graph is set up only once for a given workspace, and contains relatively a small number of nodes since the free space of OCA is represented by a single node.

[12]Note that the number of entrances is equal to that of the obstacle polygons defining the obstacle-cluster convex hull.

4.4 Applying VCDP Algorithm

With the graph and the supplementary information mentioned above, an optimal path connecting the starting (S) and ending (E) points is derived by applying the VCDP algorithm described in Section 3. Initially, S and E are identified in the graph. Then, a direction of each edge is determined as follows. Starting from the node that represents E, all the arcs connected to the node must come "into" the node. Then, for each node whose one arc has an outgoing (incoming) direction, all the other arcs connected to the node must be incoming (outgoing). Note that some arcs representing the entrances may have bidirections. Based on the directed graph obtained as above, sets of ordered barricades are determined. If either S or E is inside the OCA, then ordered sets of barricades within the OCA are obtained from the supplementary information. These sets will be added to the ordered sets of barricades outside the OCA to form complete sets of barricades. Fig. 19 shows the directed graph and four sets of barricades with the same starting and ending points in Fig. 20.

To determine an optimal path, it is necessary to evaluate every ordered set of barricades by computing the path costs with the VCDP algorithm. Note that the number of ordered sets of barricades is approximately greater than that of channels by one, since there are two sets of barricades in the corridor area. Each set of barricades determines a channel and its associated CLP. The local optimal path (LMCP) is derived by applying the VCDP algorithm. The required computation time can be reduced by ordering barricade sets to be evaluated in such a way that the set with the least number of elements is evaluated first, that with the second least number of elements next, and so on.

Figs. 20(a)–(c) show the four LMCPs, one for each of the four channels, and one global optimal path (GMCP) for three different values of the weighting factor, λ, using the same example in Fig. 19. This example is simulated with 15 gates on each barricade. This also shows that the channel where the GMCP is located changes with λ. As illustrated in these figures, when the safety factor gets large, e.g., $\lambda = 3$, the GMCP tends to traverse the most spacious channel (channel 1), while the GMCP traverses in an appropriate region (channel 3) to minimize the traveling distance as the safety factor gets smaller, e.g., $\lambda = 0.1$.

As the safety factor gets large, the LMCP tends not to deviate too much from the CLP to minimize the safety cost, especially when the LMCP traverses through a narrow-channel region (see LMCP II and III in Fig. 20a). As a result, the LMCP bends

itself several times and yields visual corners. As mentioned earlier, these visual corners disappear in the mid-range values of the weighing factor (Fig. 20b). On the other hand, when the safety factor gets small, the LMCP will be made up of straight-line segments to minimize the cost associated with distance (Fig. 20c). Note that a bend around the beginning part of LMCP II in Fig. 20(c) is due to the discretization of barricades.

5 Concluding Remarks

In this paper, a new path planning problem with both distance and safety is considered. Incorporation of robot safety into path planning is practically important not only because of the uncertainties in the robot dynamics during path execution, but also because of the inaccuracy in the geometric modeling of obstacles. Another way to view the inclusion of robot safety is to remedy the potential problems that the shortest path has: it requires a robot to traverse too close to obstacles and make very sharp turns. By considering robot safety during path planning, more realistic paths can be obtained, thus reducing/removing the difficulties during path execution caused by, for example, the uncertainties in the robot dynamics.

The VCDP algorithm is developed to solve the above path planning problem. It is computationally efficient and yields a near-optimal path in closed form for the channel characterized by the barricades and CLP. The behavior of the VCDP solution has been examined through various numerical examples. As the safety factor increases, the solution path gets closer to the safest path, CLP, while it gets closer to a SP as the safety factor decreases. For the mid-range values of the safety factor, the solution path is smooth and traverses between the two extreme paths, i.e., CLP and SP. Thus, the VCDP algorithm has high potential use for obtaining a more realistic path by selecting an appropriate value of the safety factor. The number of gates on each barricade (i.e., M) determines the accuracy of the solution and the amount of computation. As the number of gates increases, the more accurate solution will result at the expense of computational costs. Our experience with this optimization algorithm shows that $M \approx 20$ is usually more than sufficient. The VCDP algorithm is also shown to be readily extensible to a class of 3D problems.

The VCDP algorithm is applied to solve the path planning problem by representing the free space as channels. Channel boundaries are determined by convex hull operations, and the channel parameters: CLP and barricades for the main road of each channel are

derived by decomposing the concave polygon into non-overlapping convex polygons. Path planning is simply an application of the VCDP algorithm with the set of barricades identified by forming a directed graph for the decomposed workspace.

Our path planning approach has several advantages: a) the workspace is represented in compact form by a few nodes (since the obstacle-cluster area is represented by only one node) in the graph, b) only the "main road" part of each channel is considered for path planning, c) the configuration of the free space is concisely described by a set of barricades, d) the graph representing the workspace is formed only once and can be repeatedly used for the entire path planning, and e) the directed graph reduces the required search effort significantly. Note that the path planning method in Section 4 can be used independently of the first half of this paper; one can use it for finding a path avoiding collision with obstacles generously. Such a path can be formed by connecting the bisection points of the barricades retrieved from a database, which requires only a small amount of time.

Acknowledgement

The authors would like to thank Elmer Gilbert of the University of Michigan for his constructive comments on the first draft of this paper.

References

[1] M. Athans and P. L. Falb, *An Intoduction to the Theory and Its Application*, McGraw-Hill, New York, 1966.

[2] R. A. Brooks, "Solving the Find-Path Problem by Good Representation of Free Space," *IEEE Trans. Syst., Man Cybern.*, vol. SMC–13, no. 3, pp. 190–197, March/April 1983.

[3] R. Chatila, "Path Planning and Environmental Learning in a Mobile Robot System," *Proc. European Conf. on Artificial Intelligence*, Orsay, France, 1982.

[4] D. E. Kirk, *Optimal Control Theory: An Introduction*, Englewood Cliffs: Prentice Hall, Inc., 1970.

[5] D. T. Kuan, J. C. Zamiska, and R. A. Brooks, "Natural Decomposition of Free Space for Path Planning," *Proc. Int'l. Conf. Robotics Automat.*, pp. 168–173, St. Louis, March 1985.

[6] D. T. Lee and R. L. Drysdale, "Generalization of Voronoi Diagrams in the Plane," *SIAM J. Comput.*, vol. 10, no. 1, pp. 73–87, February 1981.

[7] T. Lozano-Perez and M. A. Wesley, "An Algorithm for Planning Collision-Free Paths Among Polyhedral Obstacles," *Comm. ACM*, vol. 22, no. 10, pp. 560–570, October 1979.

[8] J. Mitchell and C. Papadimitriou, "Planning Shortest Paths," Technical Report, Dept. Operations Research, Stanford University.

[9] K. G. Shin and N. D. Mckay, "Selection of Near-Minimum Time Geometric Paths for Robotic Manipulators," *IEEE Trans. Automat. Contr.*, vol. AC-31, no. 6, pp. 501–511, June 1986.

[10] J. S. Singh and M. D. Wagh, "Robot Path Planning using Intersecting Convex Shapes: Analysis and Simulation," *IEEE J. Robot. Automat.*, vol. RA–3, no. 2, pp. 101–108, April 1987.

[11] C. E. Thrope, "Path Relaxation: Path Planning for a Mobile Robot," Carnegie Mellon Univ., Technical Report CMU–RI–TR–84–5, 1984.

A Appendix A: Proof of Proposition 2

This proposition can be proved by showing a counter example. Consider two different paths connecting two sets of starting and ending points. Suppose that the optimal paths are straight-line segments, i.e., $\lambda = 0$ (Fig. 21). The equations for paths A and B are $x(\omega) = -2\omega + 2$, $\omega \in [0,1]$, and $x(\omega) = 3/4\omega$ $\omega \in [0,2]$, respectively.

Computing J for these two paths, we get $J_A = \int_0^1 (1 + (-2)^2)\, d\omega = 5$, and $J_B = \int_0^2 (1 + (3/4)^2)\, d\omega = 25/8$. The actual path lengths, $C_A = \int_0^1 \sqrt{1 + (-2)^2}\, d\omega = \sqrt{5}$, and $C_B = \int_0^2 \sqrt{1 + (3/4)^2}\, d\omega = 2.5$.

Clearly, $J_A > J_B$, but $C_A < C_B$.

B Appendix B: W-E Corner Conditions

Applying the Weierstrass-Erdmann corner conditions [4] to the functional (17) of the Path Primitive Problem:

$$\frac{\partial g}{\partial \dot{x}}\,(x^*(\omega_1, \dot{x}^*(\omega_1^-), \omega_1) = \frac{\partial g}{\partial \dot{x}}\,(x^*(\omega_1, \dot{x}^*(\omega_1^+), \omega_1), \tag{29}$$

and

$$g(x^*(\omega_1, \dot{x}^*(\omega_1^-), \omega_1) - \left[\frac{\partial g}{\partial \dot{x}}\,(x^*(\omega_1), \dot{x}^*(\omega_1^-), \omega_1)\right] \dot{x}^*(\omega_1^-)$$

$$g(x^*(\omega_1, \dot{x}^*(\omega_1^+), \omega_1) - \left[\frac{\partial g}{\partial \dot{x}}\,(x^*(\omega_1), \dot{x}^*(\omega_1^+), \omega_1)\right] \dot{x}^*(\omega_1^+), \tag{30}$$

where $g(\cdot)$ is the integrand in Eq. (17). From Eqs. (29) and (30), we get

$$2\dot{x}^*(\omega_1^-) = 2\dot{x}^*(\omega_1^+), \tag{31}$$

$$[1 + \dot{x}^{*2}(\omega_1^-)] + \lambda\,\gamma\,x^{*2}(\omega_1) - [2\dot{x}^*(\omega_1^-)]\dot{x}^*(\omega_1^-)$$

$$[1 + \dot{x}^{*2}(\omega_1^+)] + \lambda\,\gamma\,x^{*2}(\omega_1) - [2\dot{x}^*(\omega_1^+)]\dot{x}^*(\omega_1^+). \tag{32}$$

¿From Eq. (31), $\dot{x}^*(\omega_1^-) = \dot{x}^*(\omega_1^+)$, i.e., no corners exist.

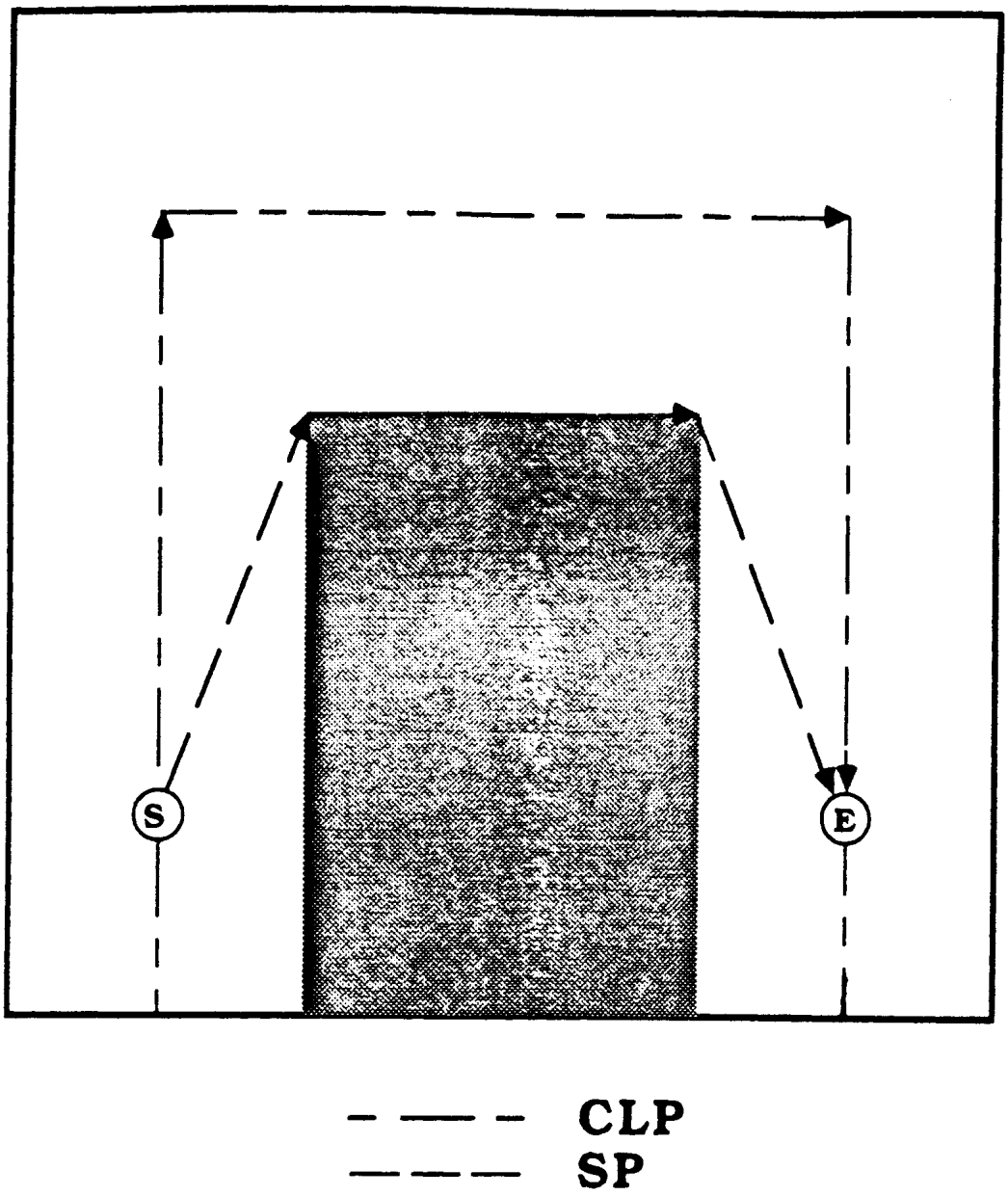


Figure 1. CLP and SP.

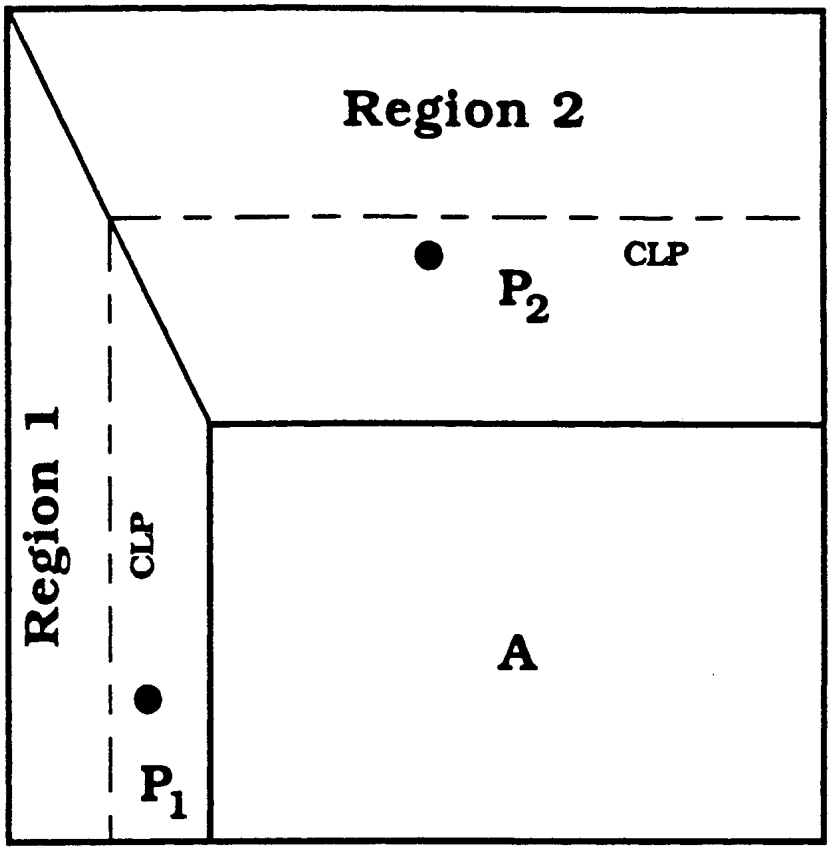


Figure 2. Two points with the same deviation from the center-line path.

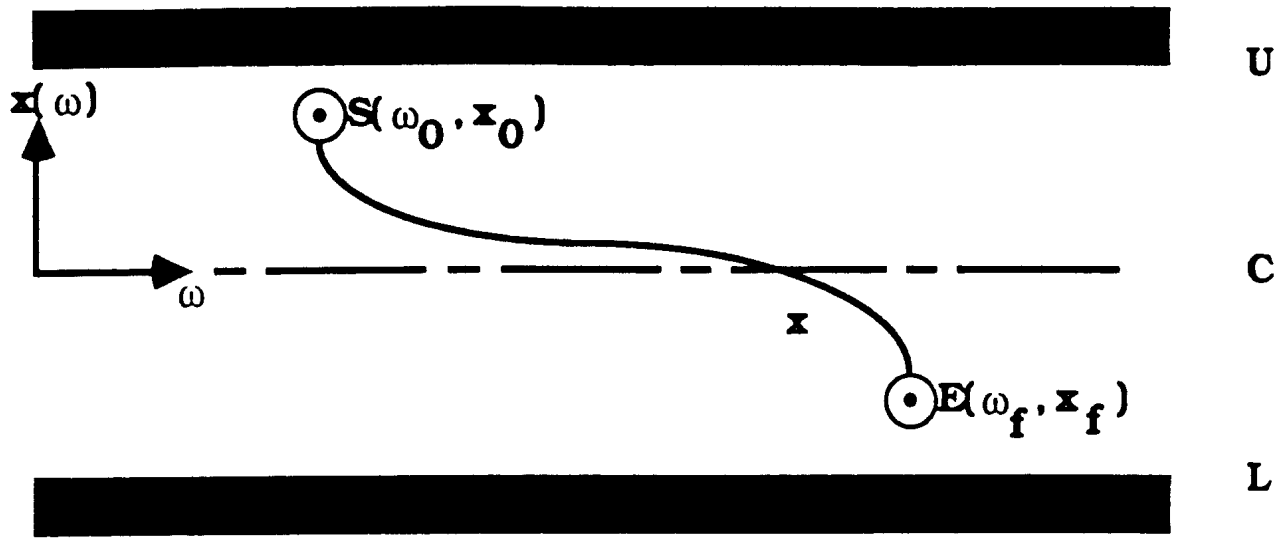


Figure 3. Problem of determining a path primitive.

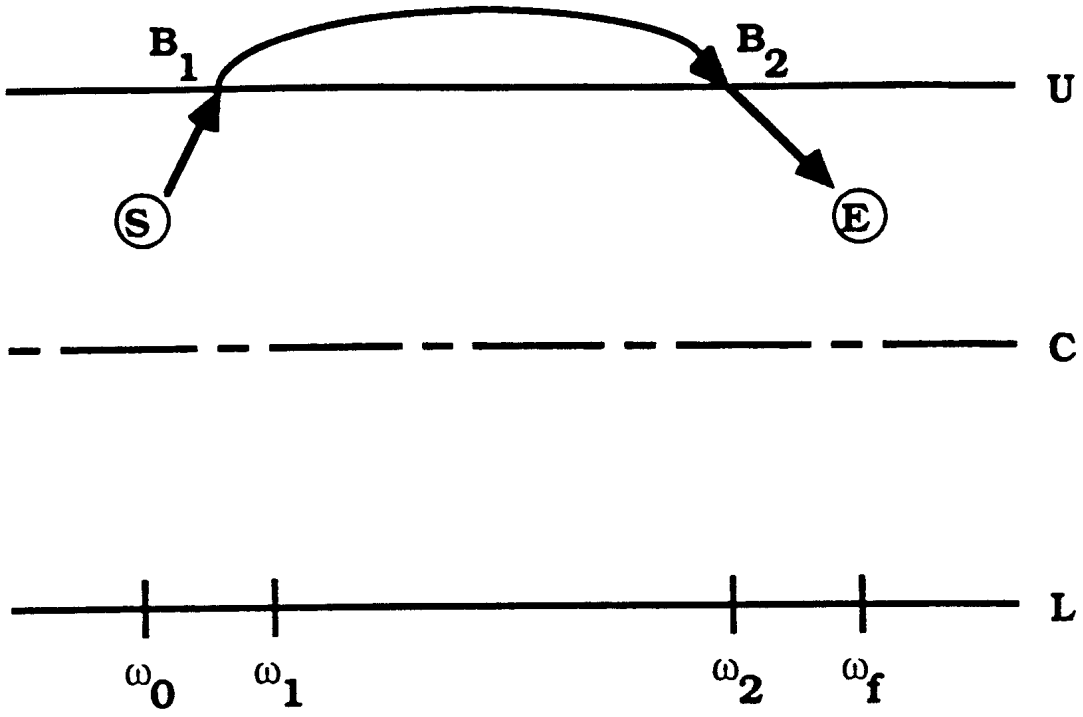


Figure 4. Break-in and break-out points.

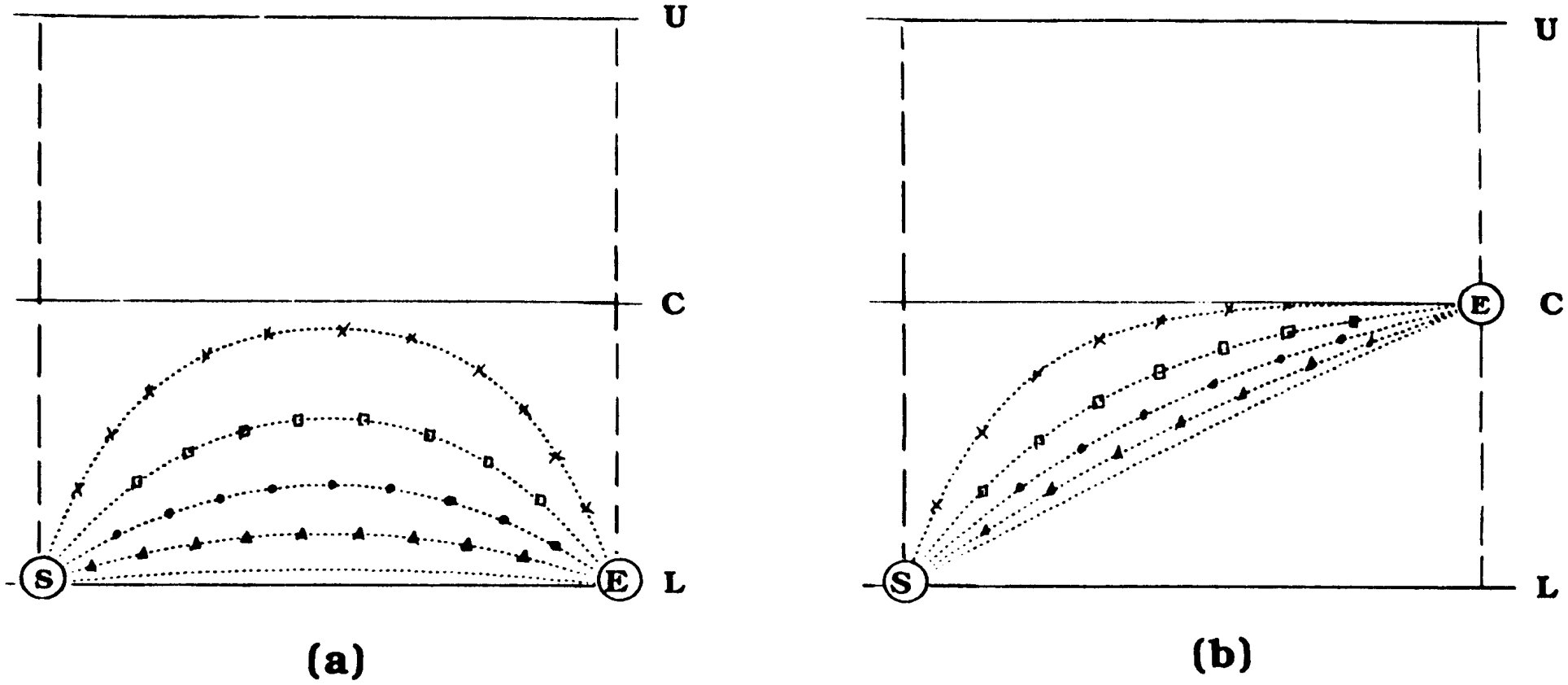


Figure 5. Optimal paths.

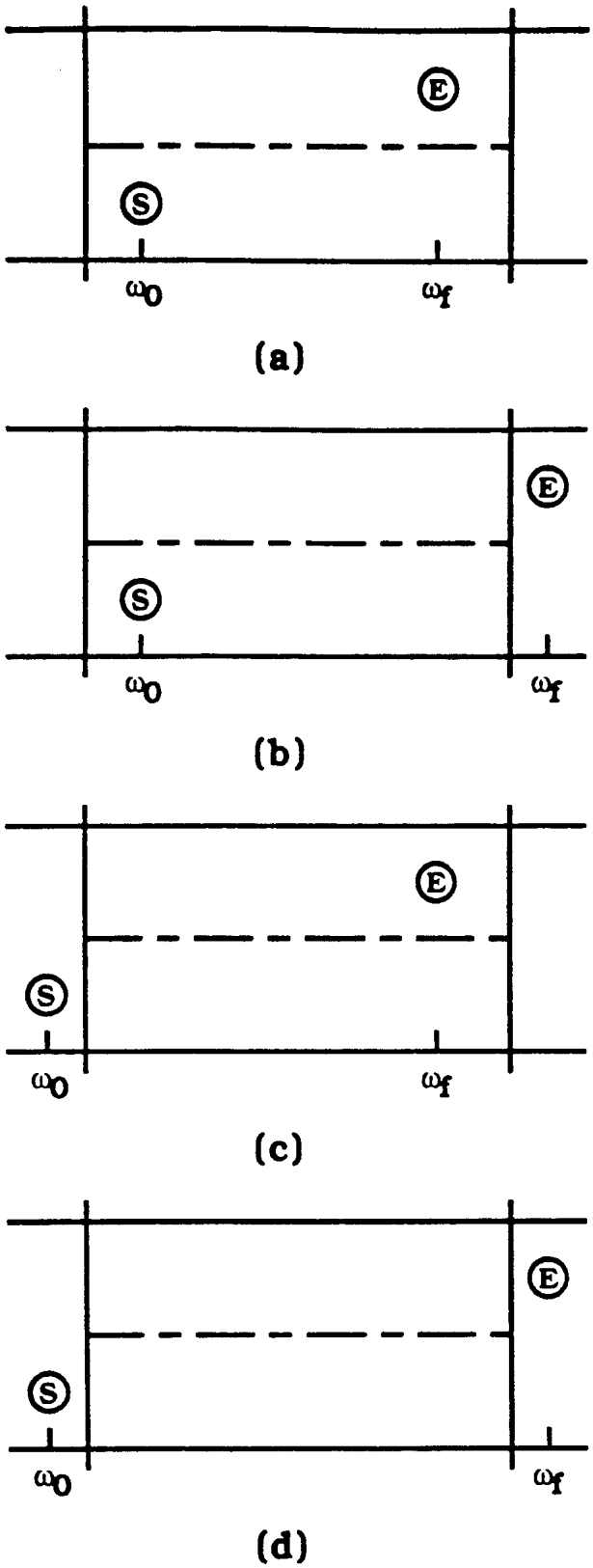


Figure 6. Four possible cases.

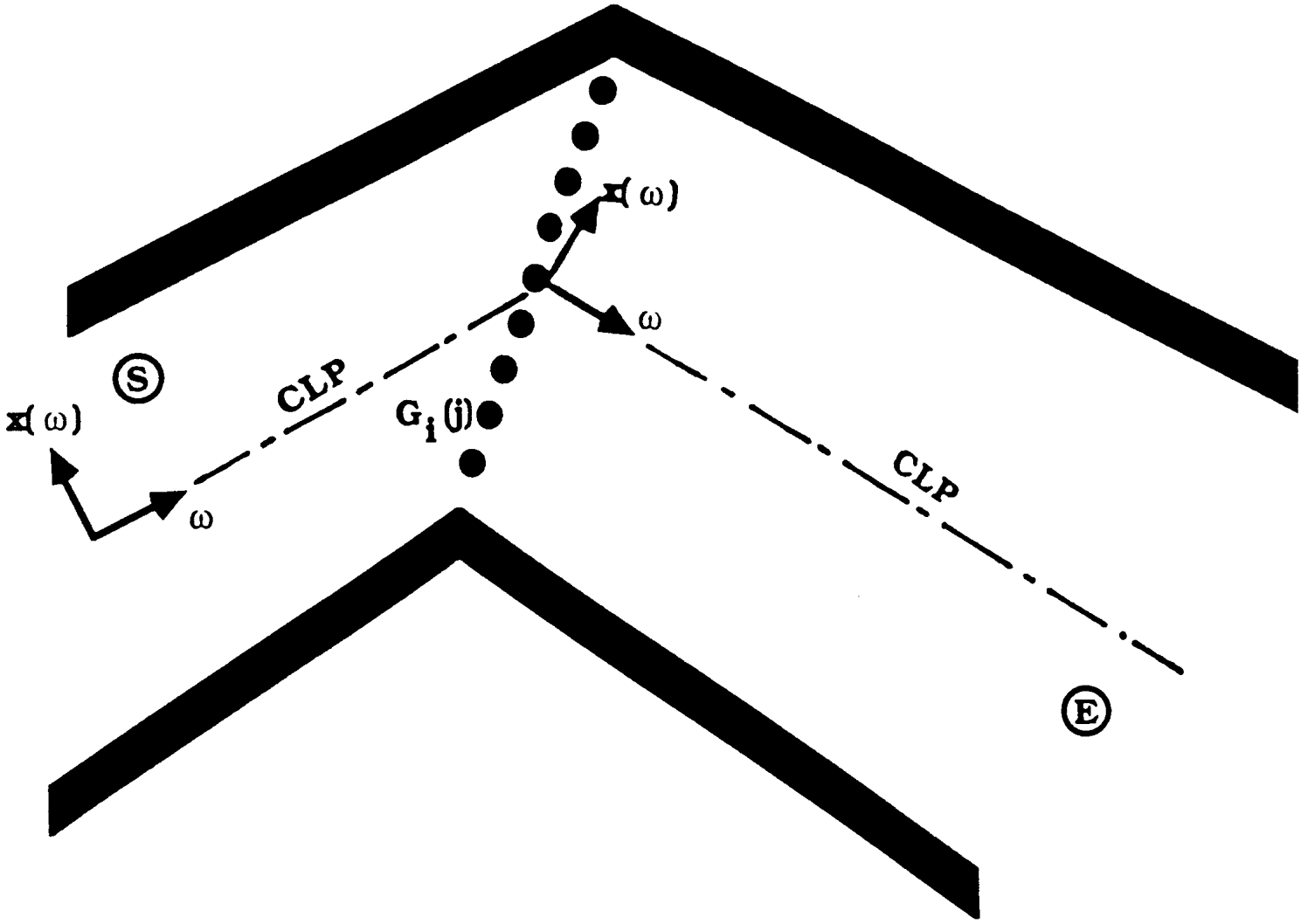


Figure 7. Dual-segmented channel.

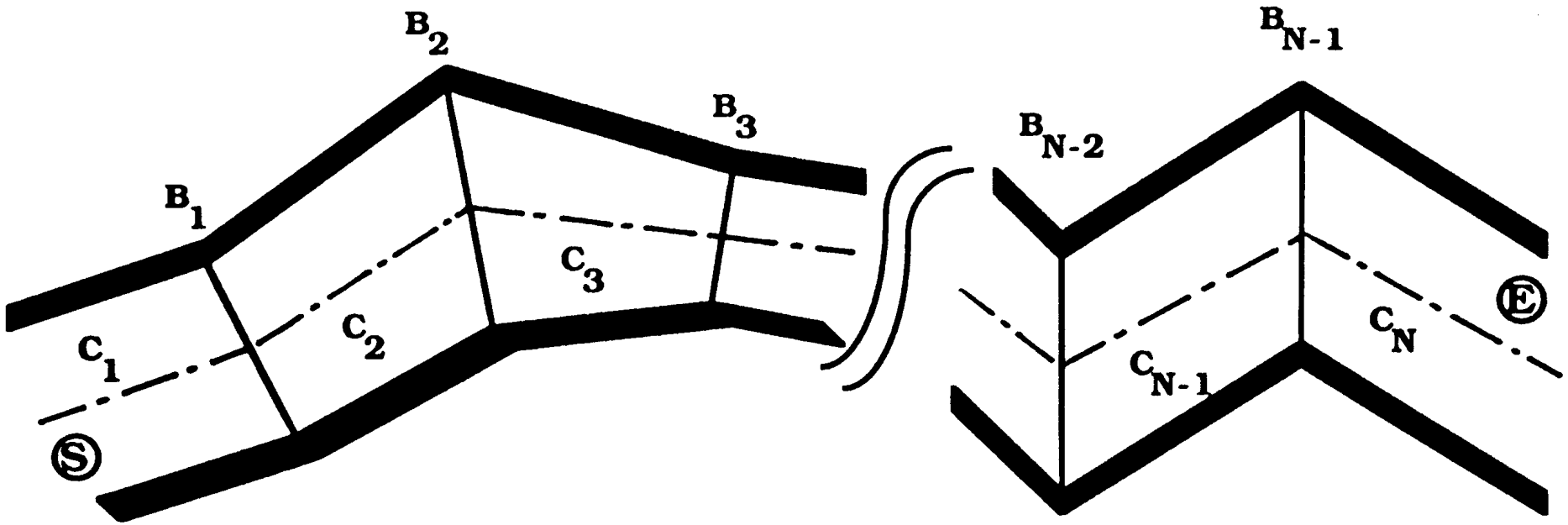


Figure 8. Multi-segmented channel.

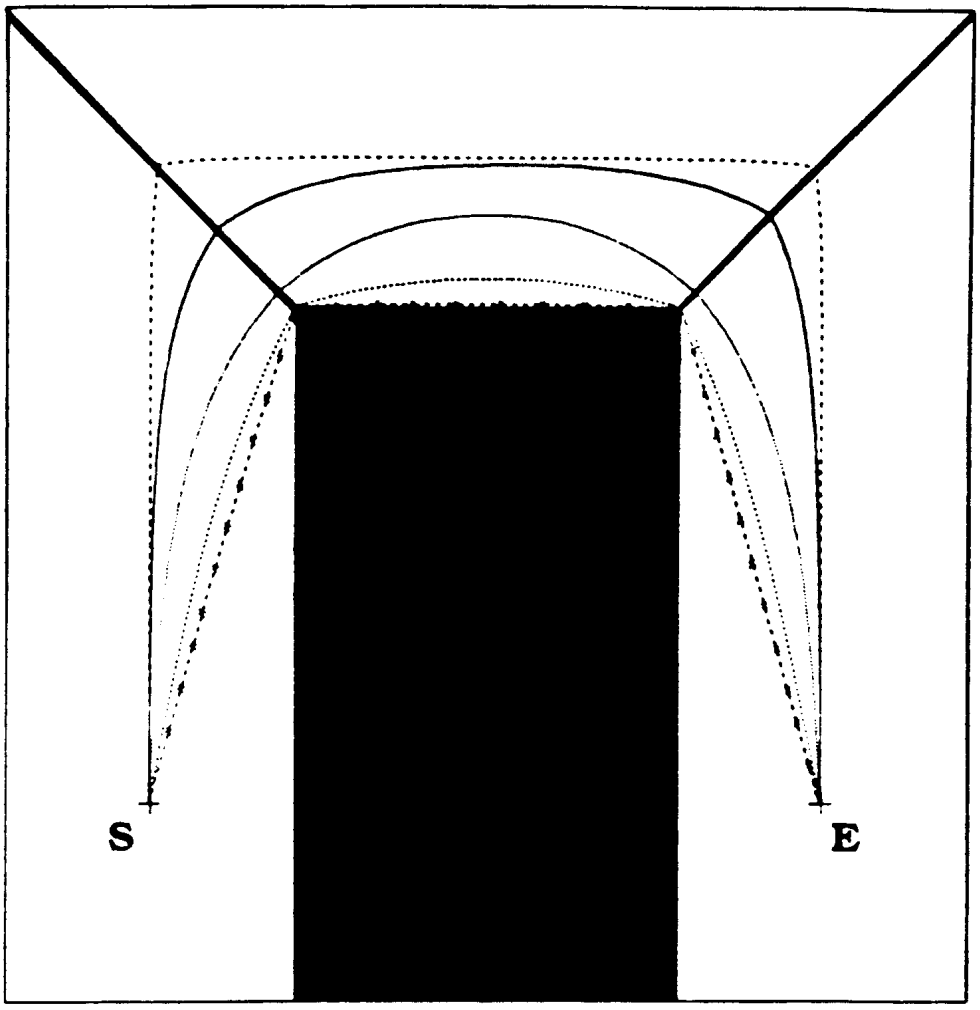


Figure 9. Example 1.

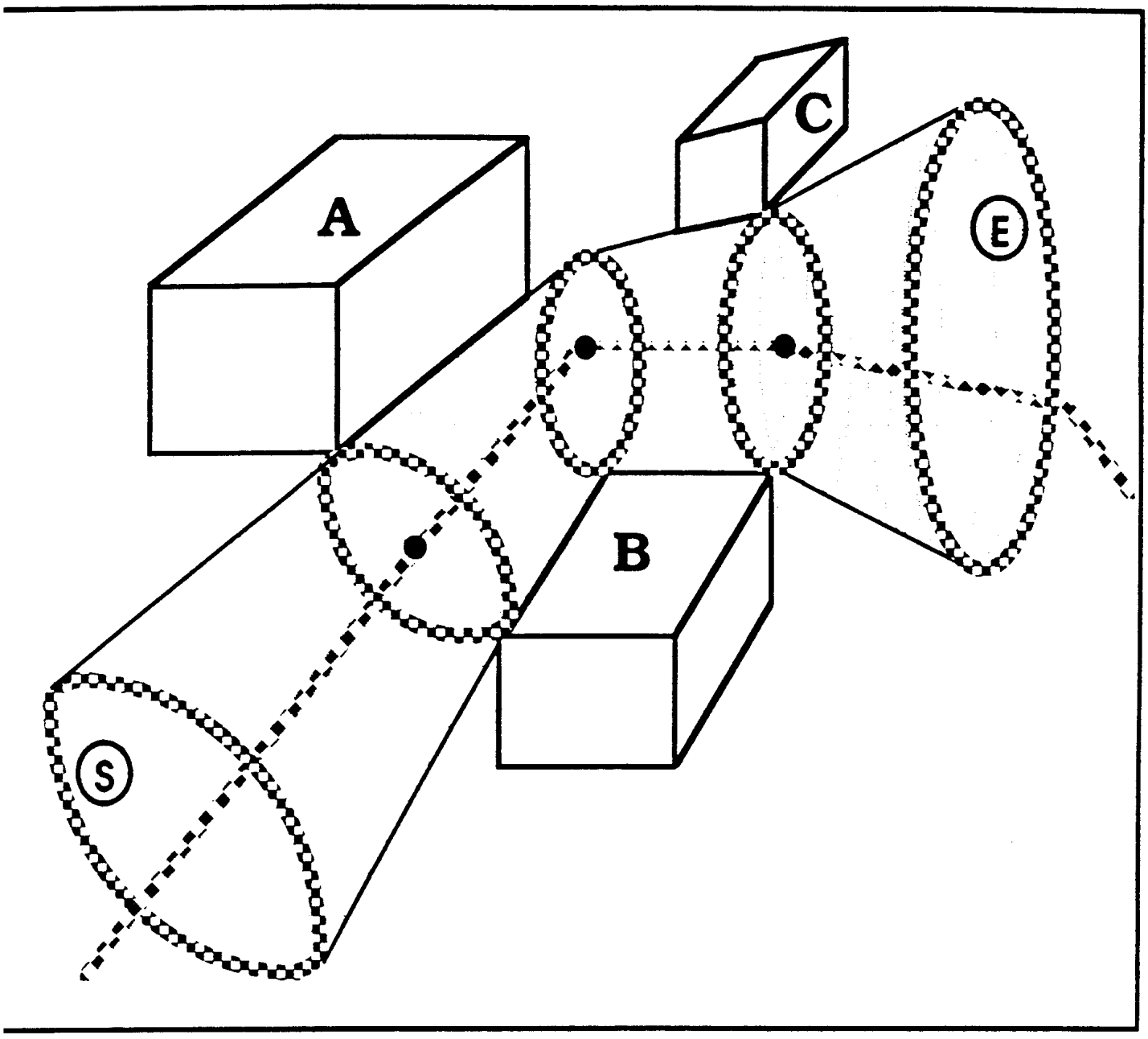


Figure 10. Articulated cylinder.

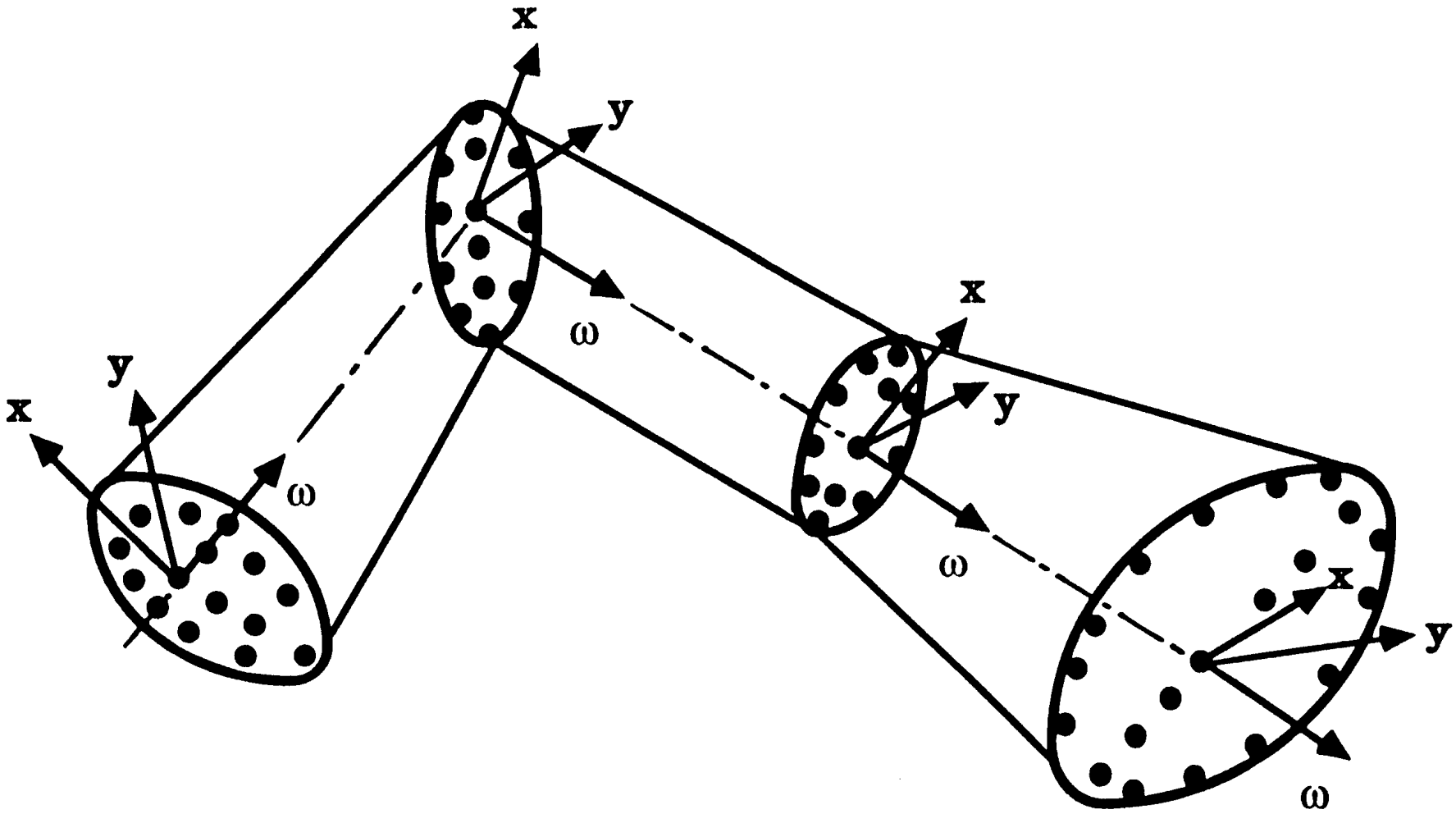


Figure 11. Barricading circles and axis nomenclature.

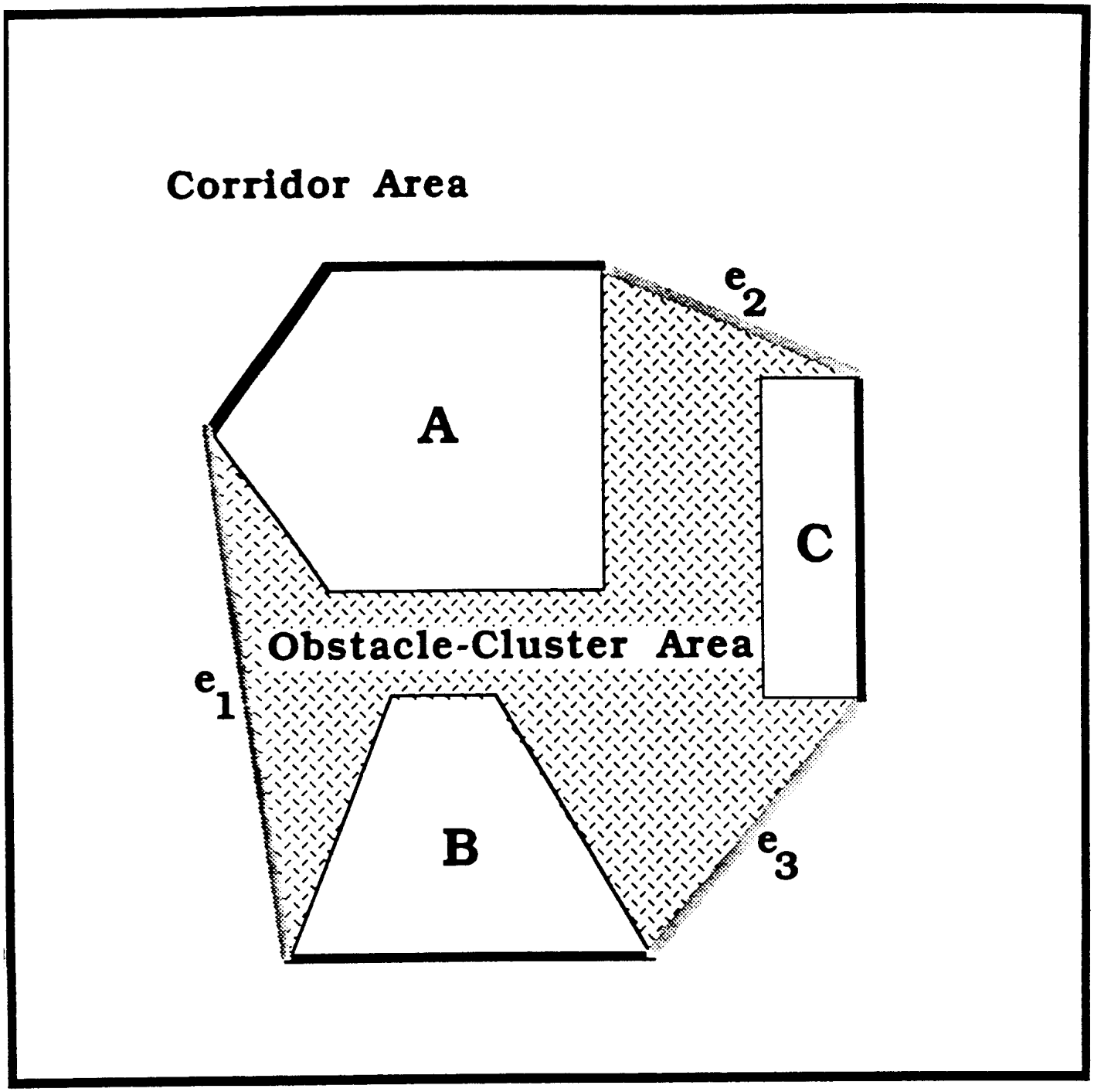


Figure 12. Two areas and entrances.

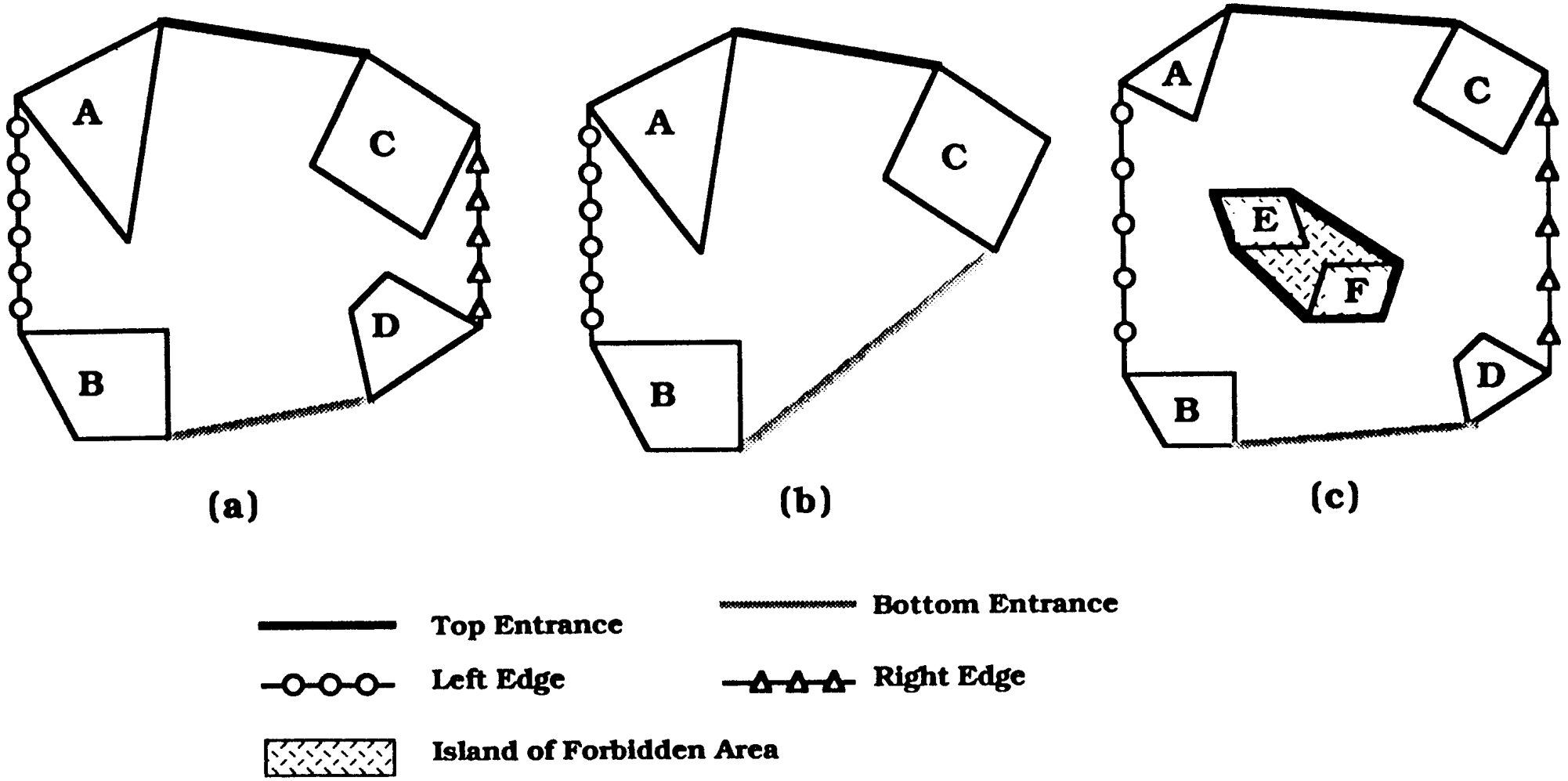


Figure 13. Entrances, edges, island of forbidden area.

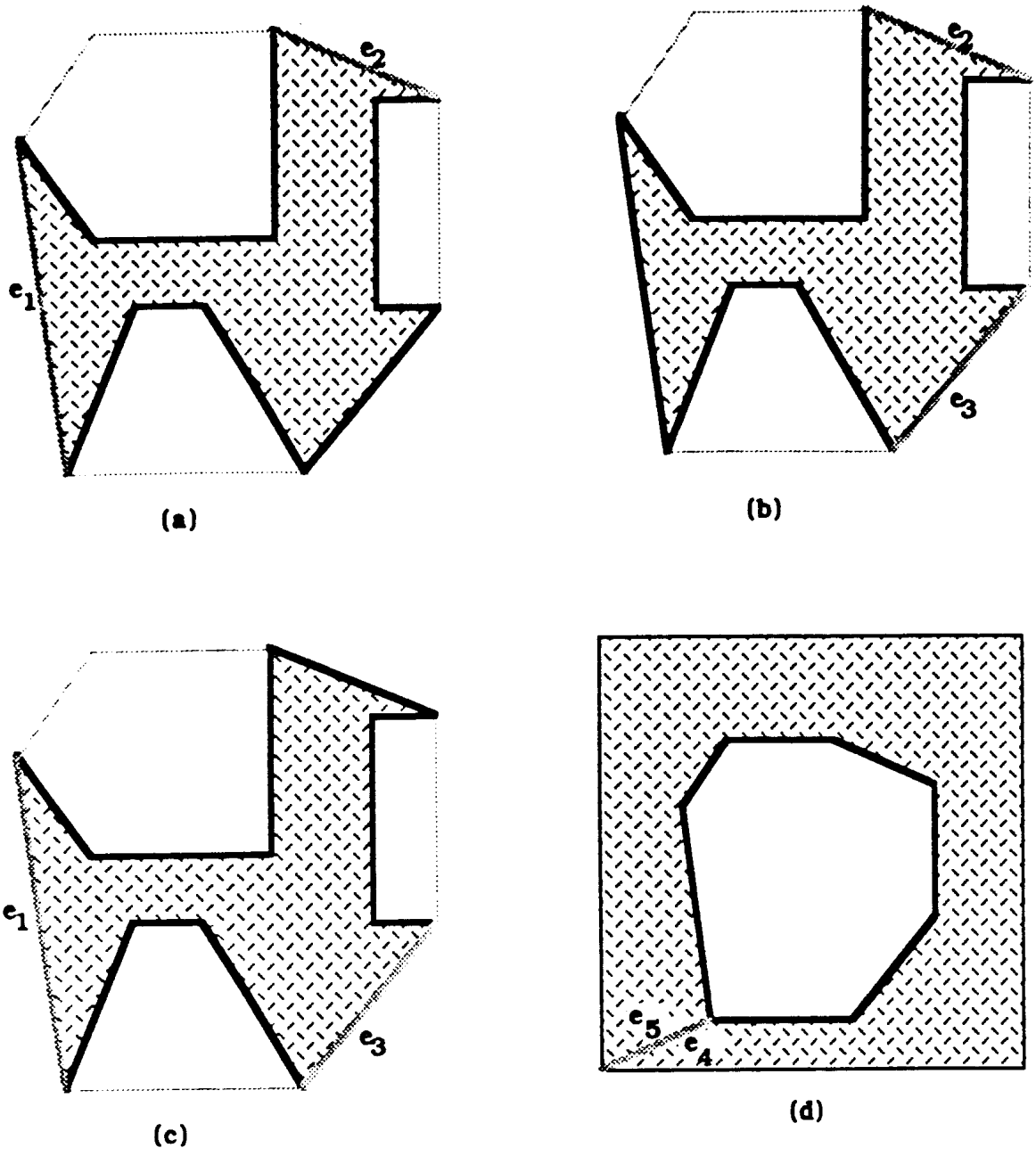
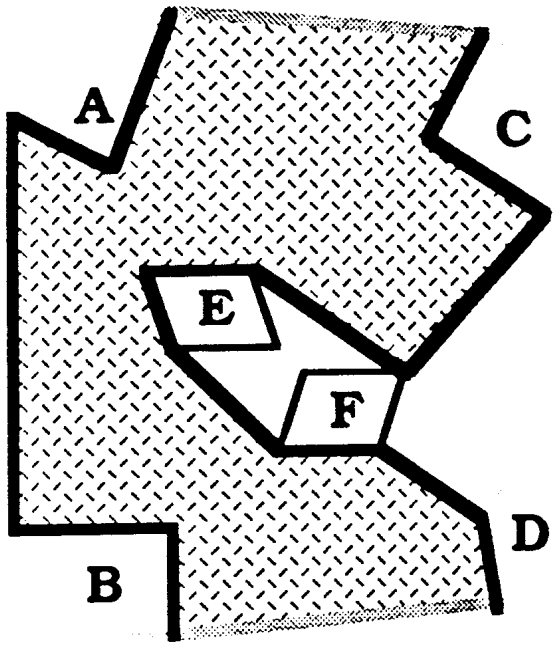
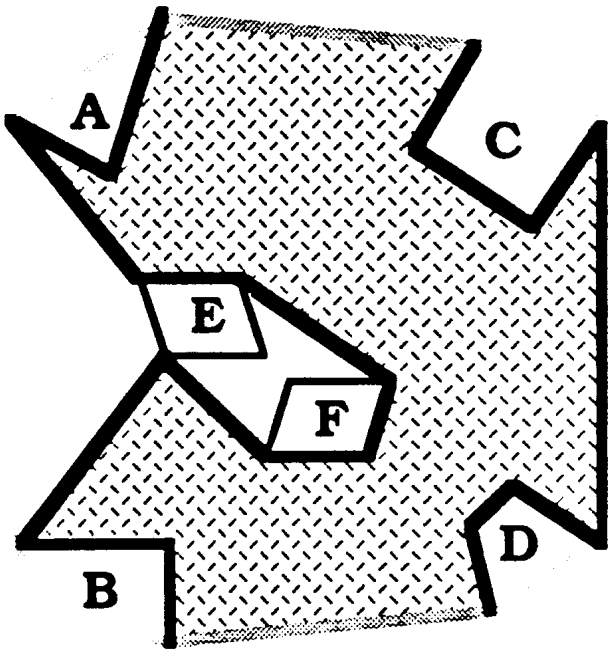


Figure 14. Channel boundaries with various entrances.

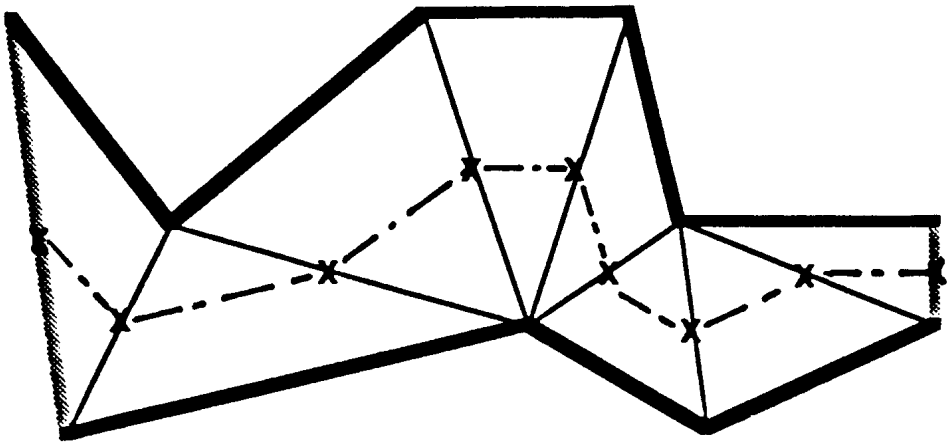


(a) Boundaries of left channel.

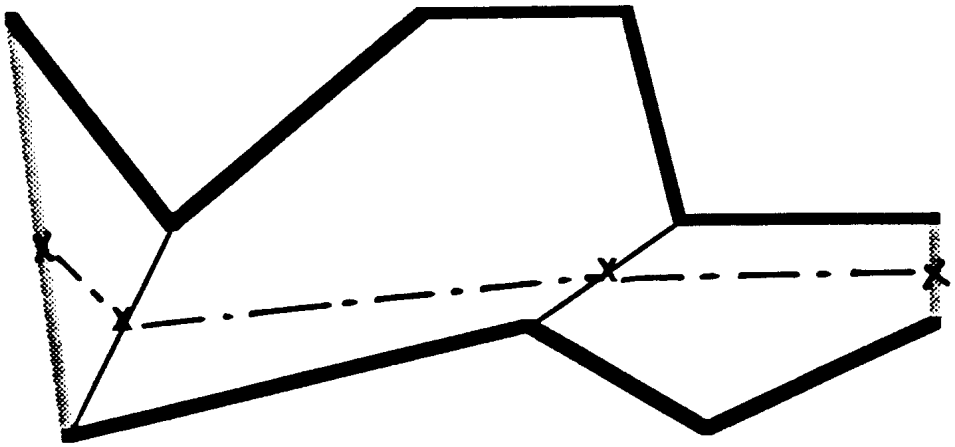


(a) Boundaries of right channel.

Figure 15. Channel boundaries for left and right channels.



(a)



(b)

Figure 16. Two CLPs.

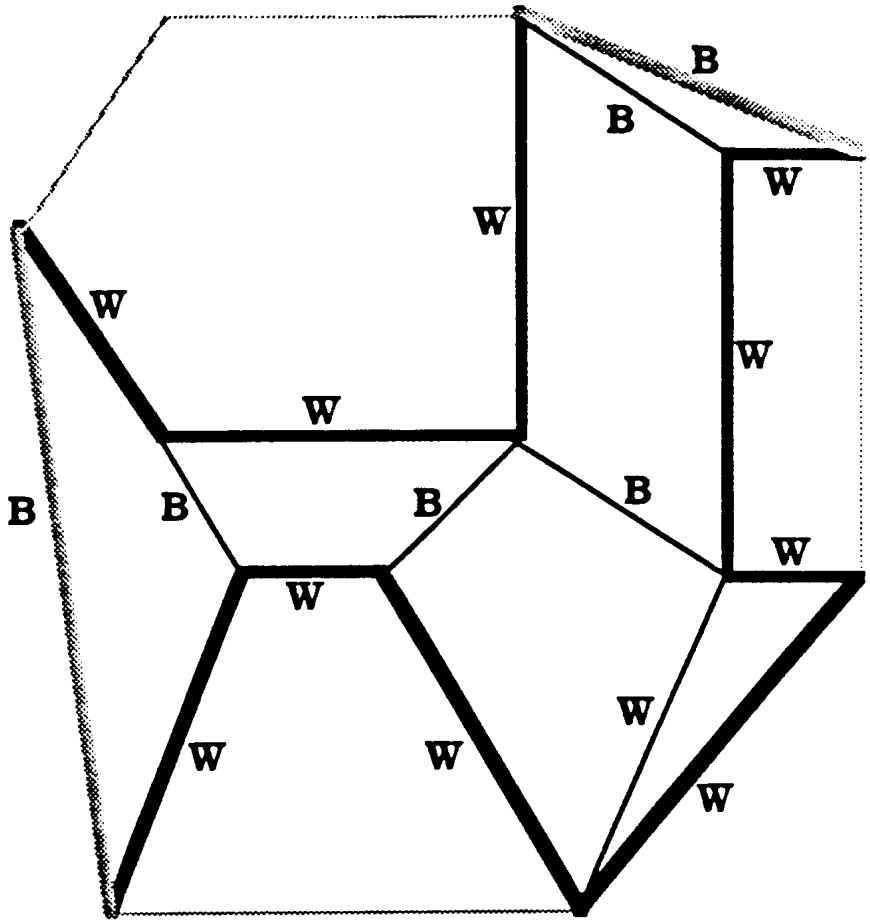


Figure 17(a). Labeling with entrances e_1 and e_2.

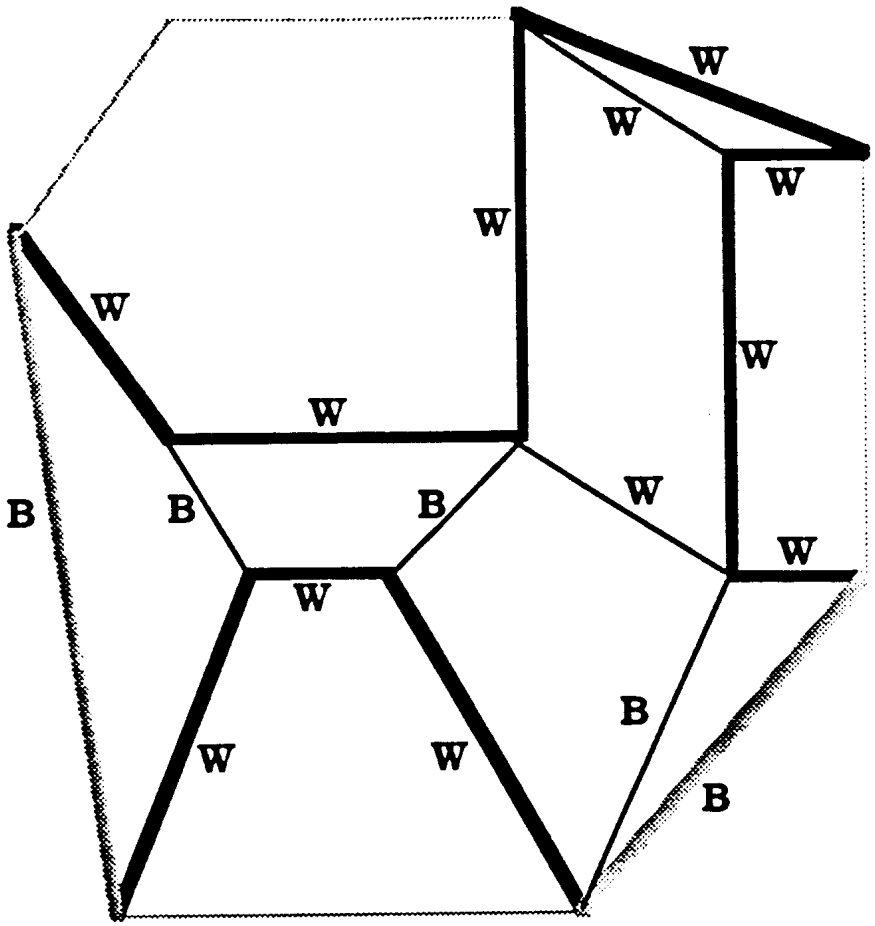


Figure 17(b). Labeling with entrances e_1 and e_3 .

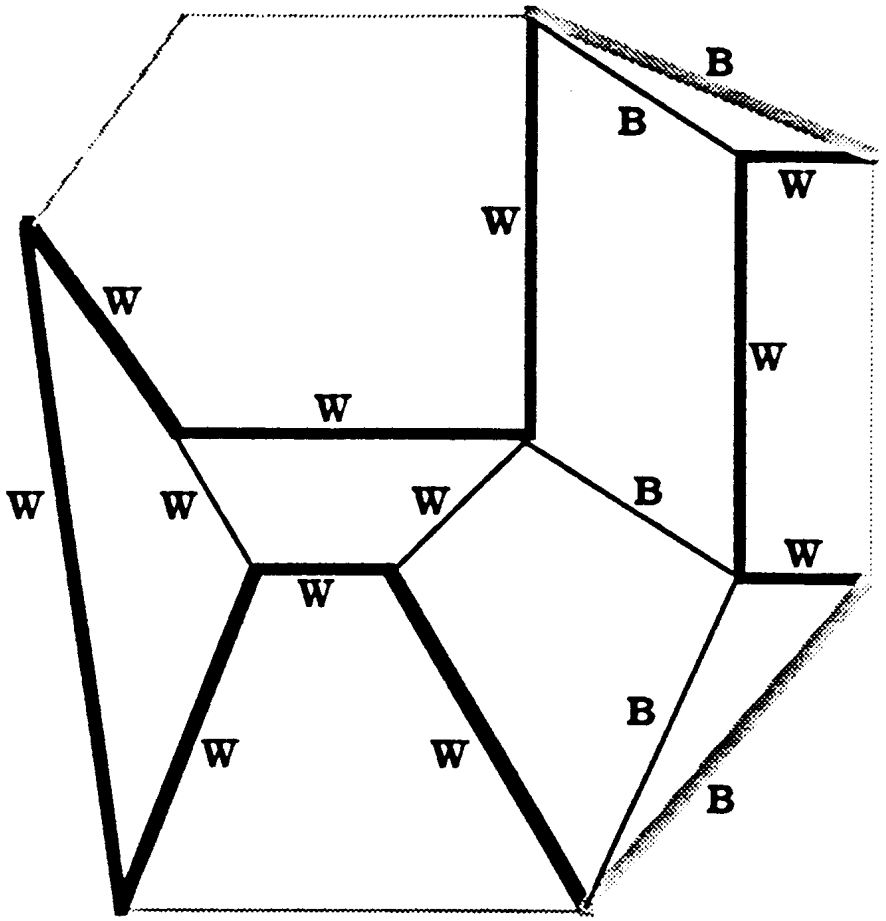


Figure 17(c). Labeling with entrances e_2 and e_3.

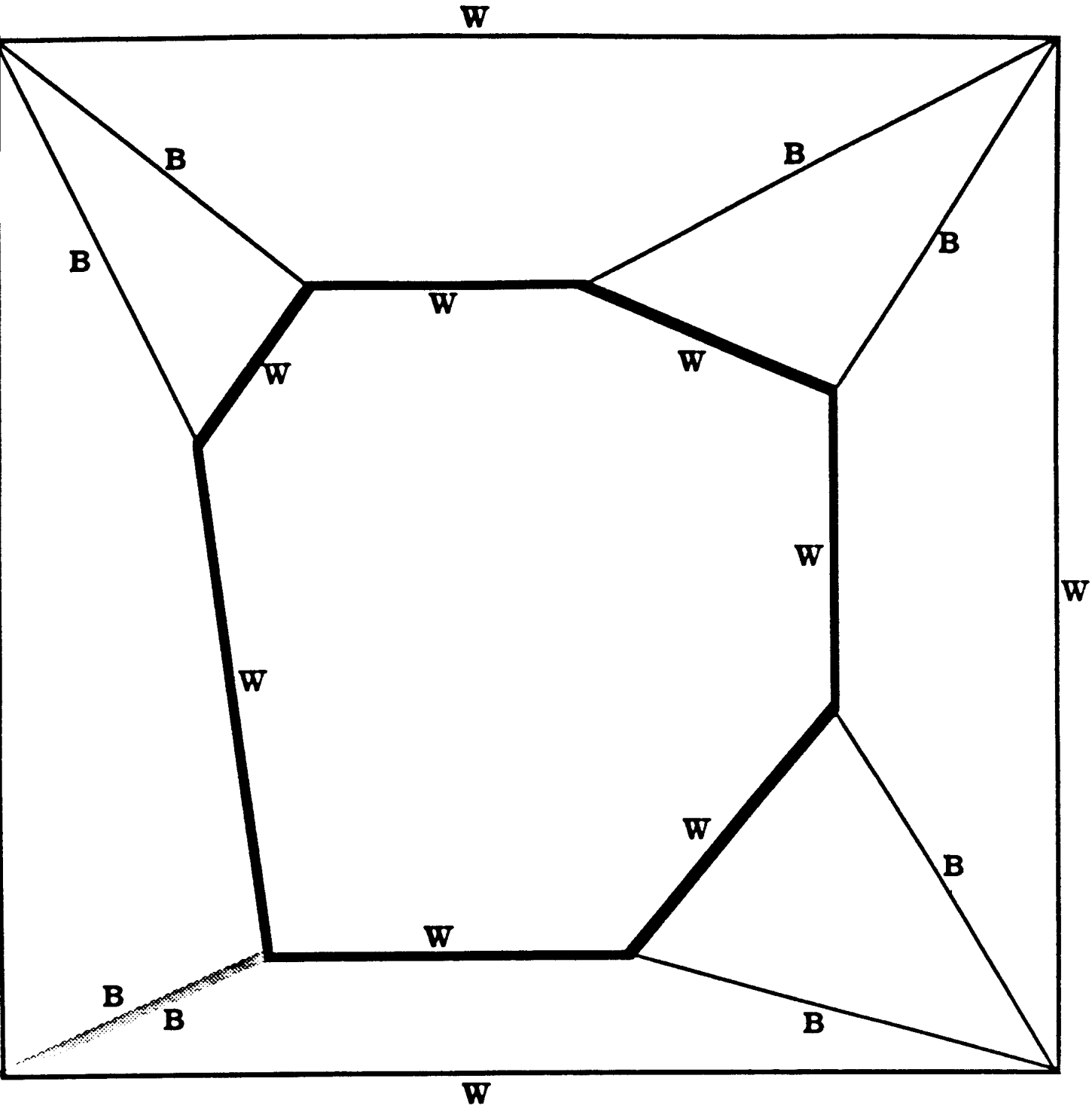
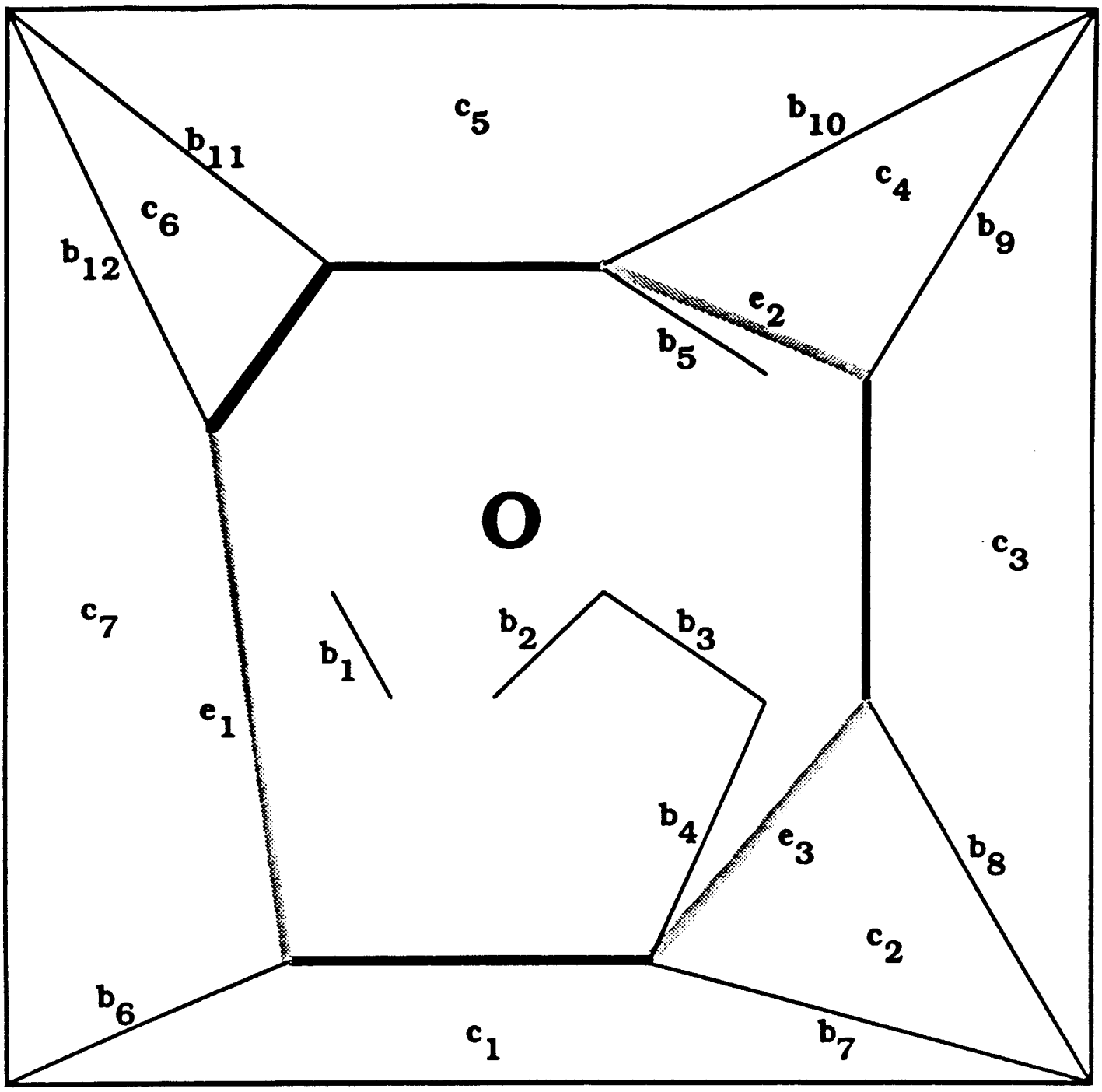
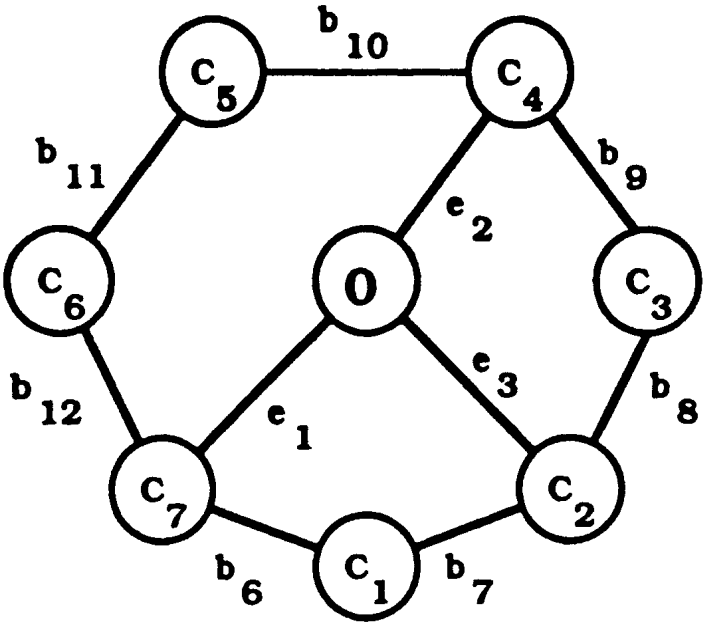


Figure 17(d). Labeling for corridor region.



(a) Notation

Figure 18. Regional graph and supplementary information.

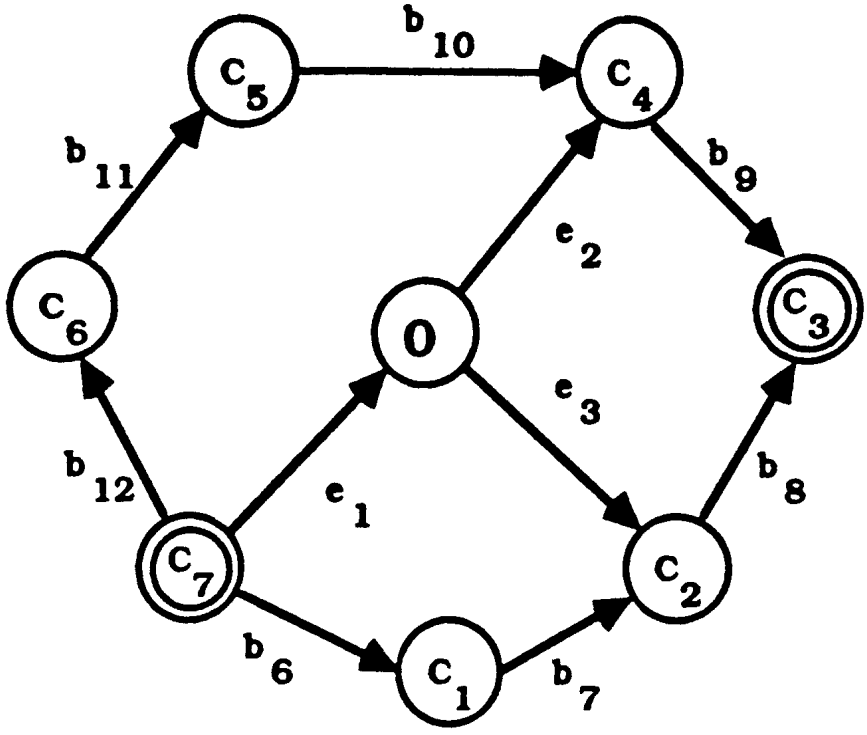


(b) Regional graph

Entrance Pair	Barricades
(e_1 , e_2)	(b_1 , b_2 , b_3 , b_5)
(e_1 , e_3)	(b_1 , b_2 , b_3 , b_4)
(e_2 , e_3)	(b_5 , b_3 , b_4)

(c) Supplementary information

Figure 18. Regional graph and supplementary information (continued).



(a) Directed graph

No.	Ordered Barricades
1	$b_{12} \cdot b_{11} \cdot b_{10} \cdot b_9$
2	$b_6 \cdot b_7 \cdot b_8$
3	$e_1 \cdot (b_1 \cdot b_2 \cdot b_3 \cdot b_5), e_2 \cdot b_9$
4	$e_1 \cdot (b_1 \cdot b_2 \cdot b_3 \cdot b_4), e_3 \cdot b_8$

(b) Four sets of barricades

Figure 19. Directed graph and four sets of barricades.

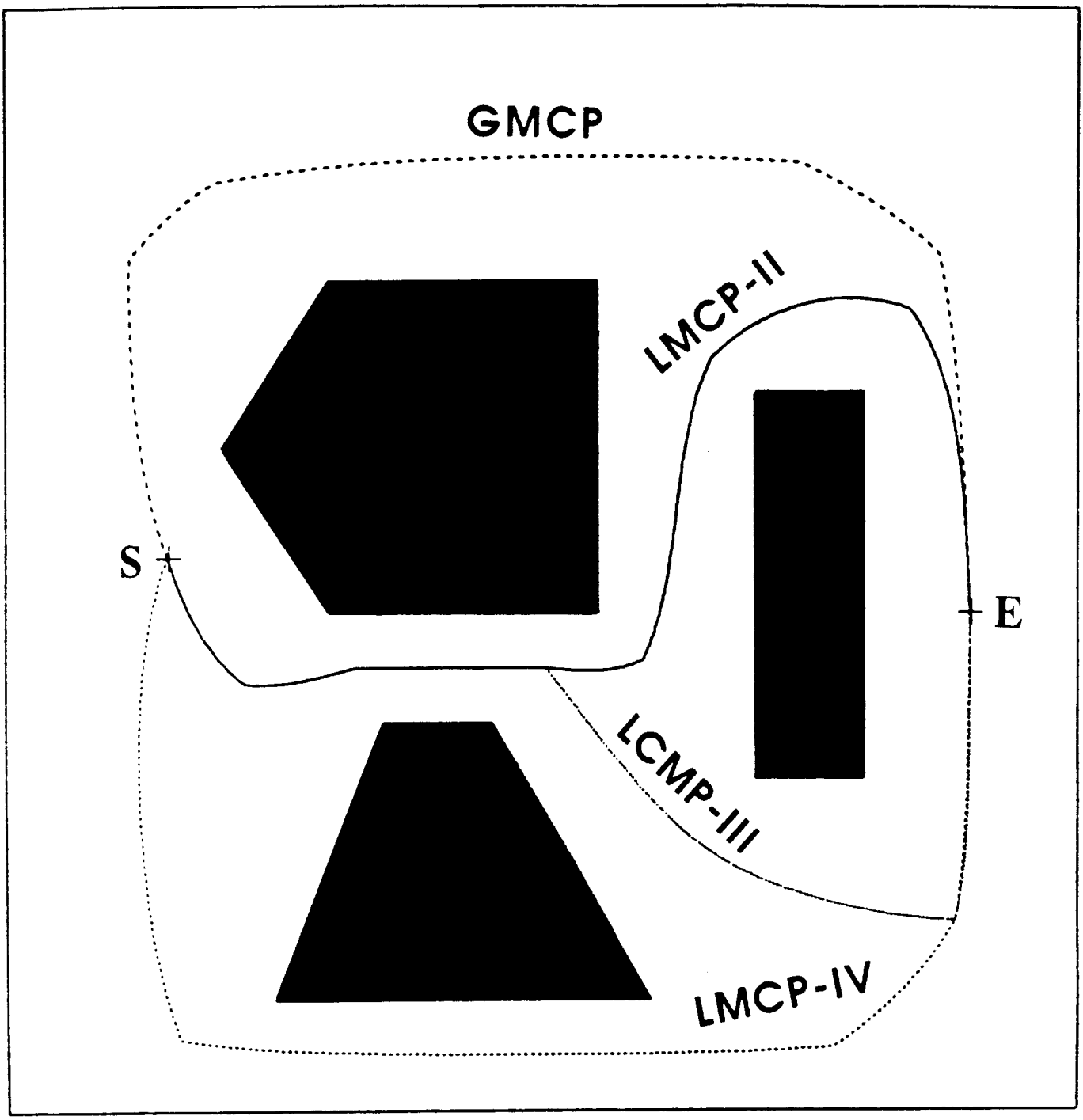


Figure 20(a). GMCP and LMCPs (λ= 4.0).

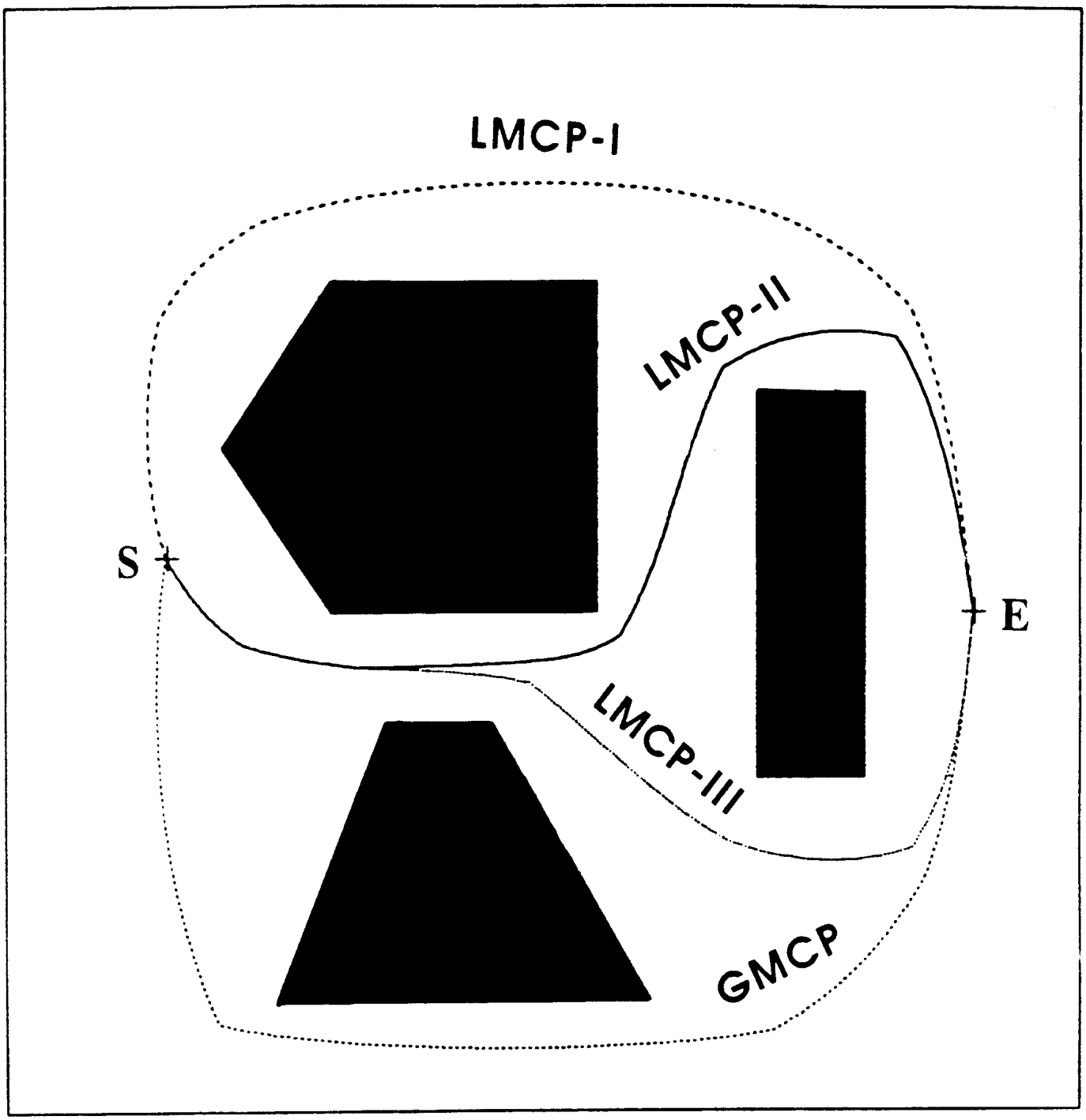


Figure 20(b). GMCP and LMCPs (λ= 1.0).

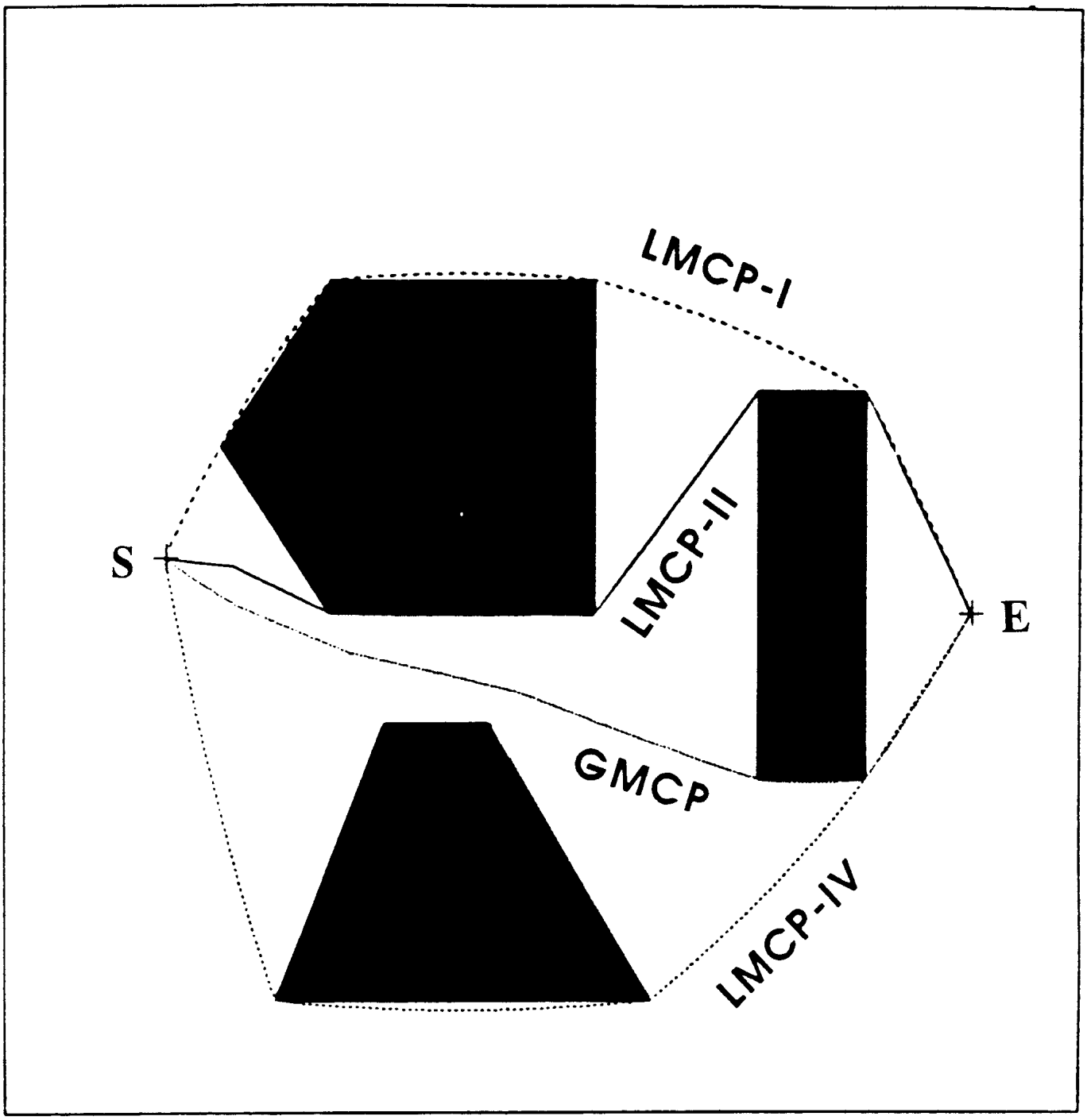


Figure 20(c). GMCP and LMCPs (λ= 0.1).

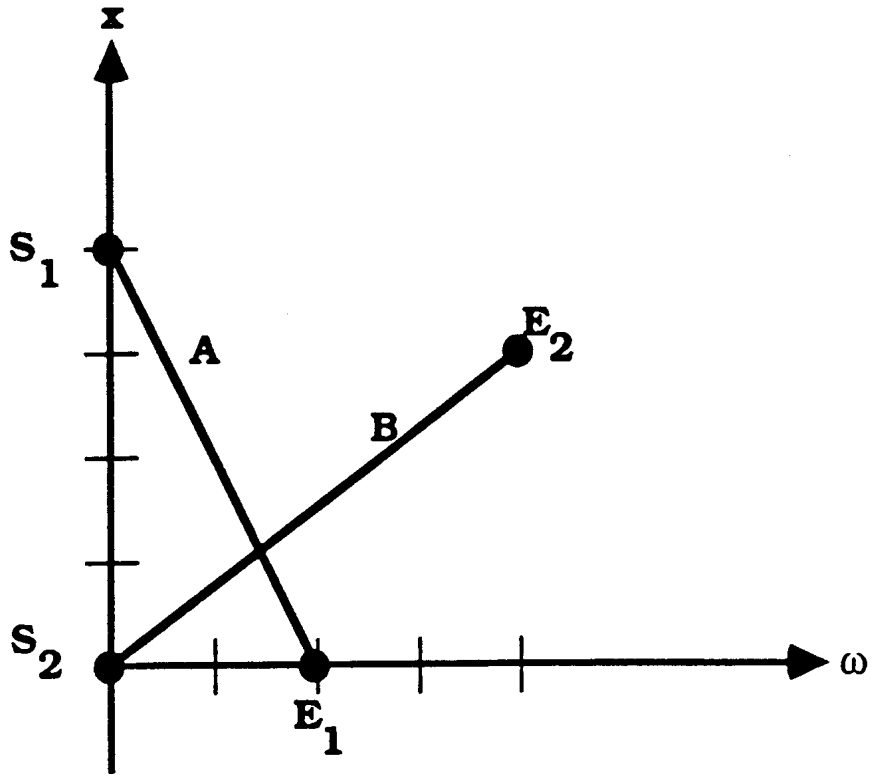


Figure 21. Two paths.