Proceedings of the 1999 IEEE
International Conference on Robotics & Automation
Detroit, Michigan • May 1999

EquiDistance Diagram – A New Roadmap Method for Path Planning

S.S. Keerthi
Indian Institute of Science
Bangalore-560012, India

C.J. Ong, E. Huang
National University of Singapore
Singapore 119260

E.G. Gilbert
University of Michigan
Ann Arbor, MI 48109

Abstract

This paper introduces a novel heuristic roadmap method for path planning, one that is inspired by the Voronoi diagram concept, but easily applicable to general configuration spaces. The roadmap is formed by connecting the local maxima of a clearance function which is defined using distance functions. Reaching the roadmap from any free configuration is achieved by applying the continuous active-set optimization algorithm to the maximization of the clearance function. Preliminary experiments with the new roadmap algorithm point to its potential utility in solving practical path planning problems.

1 Introduction

Path planning methods for determining collision-free paths for robots operating in static workspaces fall into three major groups[9]: cell decomposition, potential functions, and road map methods. In this paper we introduce a novel roadmap method. Before we discuss it, let us first describe the underlying ideas of a general roadmap method.

Let CS denote the free configuration space. A roadmap consists of V, a finite set of points in CS, and E, a finite set of feasible paths (i.e., paths that lie in CS) connecting some pairs of points in V. We will refer to an element of V as a *node* and an element of E as a *pathway*. Thus we have an abstract graph in which V defines the vertex set and E defines the edge set. We say that the roadmap is *complete* if, for each connected component of CS, the part of the graph restricted to this connected component is connected. (Hereafter we will refer to a *connected component* simply as a *component*.) To use the roadmap for path planning, we also need an algorithm $\mathcal{A}$ which, given any $x \in CS$, finds a feasible path that connects it to a point in V. Suppose a roadmap is complete and $\mathcal{A}$ is available. Then path planning between any two points x and y in CS can be done as follows. First use $\mathcal{A}$ (twice) to compute a feasible path from x to a point $v(x) \in V$, and a feasible path from y to a point $v(y) \in V$. Then use the graph defined by (V, E) to find a path (on the graph) between $v(x)$ and $v(y)$. If such a path on the graph is found then it leads to an overall feasible path between x and y. On the other hand, if a path between $v(x)$ and $v(y)$ does not exist on the graph, then, by completeness, it implies that x and y are in two different components of CS and therefore, there does not exist a feasible path between them. Different Roadmap methods differ in the way they define V, E and $\mathcal{A}$.

Next we give a quick glance of existing roadmap methods. The visibility graph[9] and Voronoi diagrams[10, 4] are limited to simple configuration spaces only. The silhouette-based methods[1, 2, 3, 12] are theoretically established to be powerful, but their complexity makes them impractical. In the probabilistic roadmap method[6] the roadmap is an extensive random network of points constructed by repeatedly generating random free configurations and connecting these configurations using some simple local planner. Algorithm $\mathcal{A}$ is implemented as follows. Given an arbitrary free configuration, x, a set of points on the network close to x are chosen and attempts are made to connect x to each of these points, using the local path planner. If all such attempts fail then random-bounce walks are tried. The method is heuristic in that completeness cannot be proven. Also, the definition and construction of the roadmap, as well as the method of connecting an arbitrary free configuration to the roadmap, are not based on systematic principles. Nevertheless, the method has been established as a useful practical tool.

We call our roadmap as *EquiDistance Diagram*, or, *EDD*, in short. It is based on a simple (yet, powerful) optimization-based algorithm, $\mathcal{A}$ for connection to the roadmap. Hence, after EDD is constructed off-line, paths between any two configurations can be computed extremely fast, on-line, using EDD. Our inspiration for EDD came from the Voronoi Diagram (VD) concept. However, EDD is very different from VD. While VD is independent of how the objects in the problem are represented, EDD is representation-dependent. (In fact, this property adds an interesting flexibility to EDD that makes it powerful.) While defining and practically constructing VD for general non-polyhedral configuration spaces is extremely difficult, EDD is easy to define and construct. As far as we know, an idea similar to EDD has not been defined and used for general path planning before. The results that we report here on EDD are preliminary; the initial findings are exciting. Detailed work on complex path planning problems is currently being done by us and these results will be presented in a later paper.

The following notations will be used in the paper. If x

is a vector, λ is a real and P is a set, $(x + \lambda P) \triangleq \{y : y = x + \lambda p \text{ for some } p \in P\}$. $|I|$ will denote the cardinality of the set I. $\|x\|$ will denote the Euclidean norm of x.

2 EDD: Definition, Properties

For the purpose of motivation, let us briefly describe Leven and Sharir's Voronoi diagram approach[10] to a two dimensional path planning problem, in which a convex polygon, P moves only by translation among convex polygonal obstacles, $\{O_i\}$ which are pairwise non-intersecting. In the base reference frame, let us represent P such that the origin is in its interior. Given a translation vector, $x \in R^2$, let us define $d_i(x)$, the one-sided growth distance between P and O_i as $d_i(x) = -1 + \min\{\lambda : (x + \lambda P) \cap O_i \neq \phi\}$. Clearly, P and O_i are non-intersecting iff $d_i(x) > 0$. Therefore the free configuration space is defined by $CS = \{x : d_i(x) > 0 \quad \forall \quad i\}$. Let us assume that CS is bounded. Let $c(x) = \min_i d_i(x)$ and $I(x) = \{i : d_i(x) = c(x)\}$. Let $V = \{x \in CS : |I(x)| \geq 3\}$ and $E = \{x \in CS : |I(x)| = 2\}$. Generically, V consists of a finite set of points in CS and E consists of a finite set of feasible paths in CS; the end points of these paths are either in V or on the boundary of CS. Let us trim E by removing from it those paths that do not have both end points in V. Leven and Sharir[10] showed that this roadmap consists of piecewise linear paths and that it is complete. They defined $\mathcal{A}$ as follows. Given $x \in CS$, find $I(x)$. If $|I(x)| = 1$ move in a straight line direction that will lead to an increase in $d_i(x)$ where $I(x) = \{i\}$, until $|I(x)| = 2$ happens. When $|I(x)| = 2$, say, $I(x) = \{i, j\}$, then move by maximizing $d_i(x)$ while maintaining the constraint, $d_i(x) = d_j(x)$. This movement leads to a point in V.

The advantages of the Voronoi roadmap defined above are: (i) the paths in E are *central*, i.e., they are located far from collision as much as possible; and (ii) $|V|$ is typically small. However, the main disadvantages are: (i) it is difficult to extend the idea to general path planning problems involving translational and rotational degrees of freedom for which the resulting CS is non-polyhedral; and (ii) even in the case of a three dimensional convex polyhedron translating among three dimensional convex polyhedral obstacles, for which CS is polyhedral, the Voronoi roadmap defined in a way similar to the two dimensional case, is not complete. Detailed algorithms have been suggested in the literature for dealing with the second disadvantage[4]. But, from a practical point of view, it is the first disadvantage that is more crucial. The EquiDistance Diagram (EDD) defined and developed in this paper is an important step in overcoming this disadvantage.

Let us formulate a general robot path planning problem. Let x denote the vector of configuration variables. Let $\mathcal{X}$ denote the configuration manifold and n be the dimension of $\mathcal{X}$, i.e., n is the number of degrees of freedom. Let P denote the physical world space in which the robot operates; typically P is some subset of either a two or three dimensional space. Let $\mathcal{L}(x) \subset P$ denote the physical space occupied, together, by all links of the robot at configuration x. Let $L_j(x)$, $j = 1, ..., l$ denote some division of $\mathcal{L}(x)$, i.e., the $L_j(x)$s are subsets of $\mathcal{L}(x)$ and their union is equal to $\mathcal{L}(x)$. For example, in the case of a manipulator, we can associate each link with one L_j and define the function of x using a representation of the link object with respect to a base reference frame and the forward kinematics associated with that link. We will assume that each L_j is a set with a nonempty interior.

Let $\mathcal{O} \subset P$ denote the physical space occupied by all obstacles together. Let $O_k, k = 1, ..., m$ denote some division of $\mathcal{O}$, i.e., the O_ks are subsets of $\mathcal{O}$ and their union is equal to $\mathcal{O}$. Note that, for a given $\mathcal{O}$, the O_ks can be chosen in many ways. We will assume that each O_k is a set with a nonempty interior.

Given two objects A and B, let $D(A, B)$ denote some measure of distance between the objects, i.e., $D(A, B) > 0$ iff $A \cap B \neq \phi$. The popular measures of distance used in the Robotics literature are the Euclidean[5] and Growth[11] distances, defined, respectively, by:

$$D_{\text{eucl}} = \min\{\|a - b\| : a \in A, b \in B\}$$

$$D_{\text{grow}} = -1 + \min\{\lambda : (p + \lambda A) \cap (q + \lambda B) \neq \phi\}$$

where p and q are reference points in the interiors of A and B, respectively. The one-sided growth distance, already mentioned in relation to Leven and Sharir's method, i.e.,

$$D_{\text{one-grow}} = -1 + \min\{\lambda : (p + \lambda A) \cap B \neq \phi\}$$

can also be employed.

Let us collect the functions, $D(L_j(x), O_k) : j = 1, ..., l, \quad k = 1, ..., m$, into a single class and name them as: $\tilde{d}_i(x) : i = 1, ..., N$, where $N = lm$. We will assume that $\tilde{d}_i \in C_1$, the class of continuously differentiable functions. (By putting appropriate smoothness requirements on the objects involved this assumption can be easily ensured. The assumption is only needed for technical reasons. If it is violated, all the algorithms relating to EDD can be extended to deal with the case, though they involve a higher computational cost.) Let us also define small clearances, ϵ_i, $i = 1, ..., N$, and require that $\tilde{d}_i(x) - \epsilon_i > 0 \ \forall i$. Define: $d_i(x) = \tilde{d}_i(x) - \epsilon_i$. Thus, CS, the free configuration space is defined by

$$CS = \{x : d_i(x) > 0 \quad \forall \quad i = 1, ..., N\}$$

We will assume that CS is bounded. Typically tranaslational degrees of freedom have physical bounds; also, if a manifold representation is used for angular degrees of freedom, they can also be bounded. Thus this assumption is quite reasonable.

To give an intuitive feeling, we first give an informal definition of EDD. Let $c(x) = \min_i d_i(x)$, the *clearance* between the robot space and the obstacle space at configuration x. Let $I(x) = \{i : d_i(x) = c(x)\}$. Consider the optimization problem,

$$\max_x \ c(x) \tag{1}$$

Let $\mathcal{A}$ denote an algorithm which, given any starting configuration, $x_0 \in CS$, generates a continuous ascent trajectory (i.e., clearance keeps increasing along the trajectory) that finally leads to a configuration which is a local maximum of the clearance function. (Note that, since $x_0 \in CS$ and $c(x)$ keeps increasing, the complete trajectory lies in CS.) The *continuous active set method*[8] is an excellent example of such an algorithm. Most often, what we find is that at such a local maximum configuration,x, we have $|I(x)| = (n+1)$. (Recall that n denotes the number of degrees of freedom.) For the sake of motivation, let us assume, for the moment, that this property holds at all local maximum configurations. An interesting and very useful associated fact is that the set, $EDD = \{x \in CS : |I(x)| \geq n\}$ defines a collection of paths that nicely connect several pairs of local maximums of $c(x)$. Therefore EDD becomes an excellent choice for consideration as a roadmap. Figures 1 and 2 give a very good example of such a roadmap. In general, EDD is not complete. So we finally add some extra paths to EDD so as to make it complete. In short, these are the basic ideas underlying EDD.

Let us now go into a formal description of EDD. It is easy to introduce a new variable, z and reformulate the optimization problem, (1) as

$$\max z \quad \text{subject to} \quad z \leq d_i(x), \ \ i = 1, ..., N \qquad (2)$$

Let $\mathcal{I}$ denote the set of all non-empty subsets of $\{1, ..., N\}$. The first order necessary condition for this problem is that there exists $I \in \mathcal{I}$ and $\mu_i, i \in I$ such that the following hold:

$$z = d_i(x) \ \ \forall \ \ i \in I \qquad (3a)$$

$$z < d_i(x) \ \ \forall \ \ i \notin I \qquad (3b)$$

$$\mu_i \geq 0 \ \forall \ i \in I, \ \ \sum_{i \in I} \mu_i = 1, \ \ \sum_{i \in I} \mu_i \nabla d_i(x) = 0 \qquad (3c)$$

where $\nabla d_i(x)$ denotes the gradient of d_i with respect to x. Hereafter we will refer to (3a)-(3c) simply as (3). Let

$$\tilde{V} = \{x \in CS : (3) \text{ holds for some } z \text{ and } \mu_i\}$$

Under reasonable conditions we can show that $\tilde{V}$ is a finite set. To do this let us take the tolerances, ϵ_i to be linear functions of x:

$$\epsilon_i(x) = \alpha_i x + \beta_i$$

Let us put bounds on these parameters:

$$|\alpha_i| \leq \alpha_{\max}, \quad \beta_{\min} \leq \beta_i \leq \beta_{\max} \qquad (4)$$

where

$$\alpha_{\max} > 0 \quad \text{and} \quad 0 < \beta_{\min} < \beta_{\max} \qquad (5)$$

Suppose ϵ is any given positive number. Then, since CS is bounded, $\alpha_{\max}$, $\beta_{\min}$ and $\beta_{\max}$ that satisfy (5) can be chosen such that the following holds: for any α_i and β_i that satisfy (4), we have

$$0 < \epsilon_i(x) < \epsilon \ \ \forall \ x \in CS$$

Thus any user-set tolerance on our" manipulation" of distance functions can be met by appropriately selecting the parameter bounds, $\alpha_{\max}$, $\beta_{\min}$ and $\beta_{\max}$. Let us say that these constants have been selected and fixed. Transversality theorem[7] can be used to show the following useful result.

Theorem 1. *For almost all choices of (α_i, β_i) satisfying (4), $\tilde{V}$ is a finite set.*

As already mentioned, for algorithm $\mathcal{A}$ we use the continuous active set method[8], which is a gradient-based ascent method for generating a trajectory that leads to a local maximum of the optimization problem, (2). It can be shown that, *for almost all starting points, $x_0 \in CS$, the continuous active set method generates a path in CS that finally terminates at a point $x \in \tilde{V}$.* This together with the fact that $\tilde{V}$ is a finite set prompts us to consider $\tilde{V}$ as the basis for defining a roadmap. Let us partition $\tilde{V}$ into the following three disjoint sets:

$$\tilde{V}_1 = \{x \in CS : (3) \text{ holds for some } z, \{\mu_i\}, \ \& \ |I| > n\}$$

$$\tilde{V}_2 = \{x \in CS : (3) \text{ holds for some } z, \{\mu_i\}, \ \& \ |I| = n\}$$

$$\tilde{V}_3 = \{x \in CS : (3) \text{ holds for some } z, \{\mu_i\}, \ \& \ |I| < n\}$$

As already mentioned, our definition of EDD is also strongly motivated by a typical observation: only in special rare conditions we come across an $x \in \tilde{V}$ such that the I in (3) corresponding to that x satisfies $|I| < n$. (For instance, in all of the examples mentioned in section 4 we did not come across even a single x with this property.) So our main concentration will be on $\tilde{V}_1 \cup \tilde{V}_2$ and how to systematically develop pathways for connecting the points in that set. Let $x \in \tilde{V}_2$. Then by omitting the optimality requirement, (3c), and tracking only (3a)-(3b) we can generate a one dimensional pathway passing through x. Suppose $x \in \tilde{V}_1$. Then by trimming I (in many ways) so that it has exactly n elements and tracking only (3a)-(3b) leads to a number of pathways passing through x. This prompts us to consider the set of pathways defined by

$$E = \{x \in CS : (3a) - (3b) \text{ hold for some } z \ \& \ |I| = n\}$$

Clearly, $\tilde{V}_2 \subset E$. The set of nodes naturally associated with E is

$$\tilde{V}_4 = \{x \in CS : (3a) - (3b) \text{ hold for some } z \ \& \ |I| > n\}$$

Clearly, $\tilde{V}_1 \subset \tilde{V}_4$. Transversality theorem[7] can be used to establish the following result that describes the relation between E and $\tilde{V}_4$.

Theorem 2. *For almost all choices of (α_i, β_i) satisfying (4), the following hold. (i) E is the union of a finite number of one-dimensional manifolds (pathways). (ii) $\tilde{V}_4$ is a finite set. (iii) $\tilde{V}_4$ does not have any element x for which $|I| > (n+1)$. (iv) At each $x \in \tilde{V}_4$, exactly $(n+1)$ pathways of E terminate; and, if I is the index set corresponding to x, then these pathways are defined by removing one element at a time from I and following (3a) and (3b).*

If we define

$$V = \tilde{V}_2 \cup \tilde{V}_4$$

then V and E define a powerful roadmap for use in path planning. Trimming off pathway may also be done to simplify the network. For example, we can trim off any pathway (or part of a pathway) which directly connects a point on the boundary of CS (i.e., a collision point) to a point in $\tilde{V}_2 \cup \tilde{V}_4$. After these trimming operations, the network consists of: V, a set of points which necessarily contains $\tilde{V}$ in it and also contains some points of $\tilde{V}_4$; and, E, a set of pathways connecting some pairs of elements of V. We refer to $V \cup E$ as the EDD.

As in the two dimensional example discussed at the beginning of this section, V and E define an abstract graph. This graph, in association with algorithm $\mathcal{A}$ defined by the continuous active set method, helps us do path planning. In general, EDD is not complete. However, EDD is very valuable because: (i) the number of components of EDD is usually small; and (ii) the region of CS attracted by each component of EDD (via application of $\mathcal{A}$) is large. Because of these properties, adding extra pathways to EDD so as to make it complete is not very hard. We approach this problem of incorporating completeness using heuristic schemes that will be discussed in the next section.

3 Construction of EDD

The very definition of EDD suggests a simple algorithm for constructing an EDD component. Suppose we are given one $x_0 \in CS$. Then we can apply $\mathcal{A}$ to reach a point, $x \in \tilde{V}$. Let I denote the index set of distance functions active at x. If $|I| < n$ then $x \in \tilde{V}_2$ and so it is an EDD component. If $|I| = n$ then we leave out (3c) and track the pathway defined by (3a) and (3b) which passes through x. If $|I| = (n+1)$ then we define $(n+1)$ pathways at x by tracking (3a)-(3b) after leaving out one element of I at a time. As we do the pathway tracking we look for encounters with points of V; this can be done by using a numerical curve tracing algorithm with root finding capability. When a point of V is encountered, it is expanded in a similar way as at x. The process is continued until all elements of V that have been encountered are fully expanded. We will not go into a detailed description of the procedure as it is rather obvious. Note from the way the above computations proceed, that, such a construction of an EDD component is very suited for sensor-based computations where only local information is available in any situation.

Let us now consider the issue of adding extra pathways so as to ensure completeness of EDD. The first problem is to determine the various components of EDD. As we already discussed above, once we get a point on an EDD component, it is quite easy to numerically track all details of that component (nodes, pathways etc.). How do we obtain at least one point on each EDD component? Let EDDc be one component of EDD. Let X_c denote the set of all points of CS which will lead to a node of EDDc when $\mathcal{A}$ is applied. We can view X_c as the region of attraction for EDDc. As remarked earlier, typically, the number of components of EDD is small and so X_c covers a sizeable fraction of CS. Hence, even if we put a coarse grid of points in the configuration space, there is an excellent possibility that there is at least one grid point that belongs to each component of EDD. Therefore we place a coarse grid of points in the configuration space, and determine the EDD component associated with each one of these points which lie in CS. (For efficiency reasons, we incorporate simple checks to find out if a point leads to a component that has already been tracked.) The above described scheme has worked well on all examples that we have tried. Currently, we are also working on other elaborate schemes for determining all EDD components.

Once the various EDD components have been found, the remaining job is to connect them. Note that, if CS itself is disconnected, then it is impossible to connect a component of EDD that lies in one component of CS with another component of EDD that lies in a different component of CS. We proceed as follows. We take one pair of EDD components at a time and search on these components to find one point in each of these components such that these points are close to each other. After doing this for every pair of EDD components we rank order the pairs of points, thus found, according to closeness. Then we take one pair at a time, in that order, and make attempts to connect the two points in the pair by a pathway. We can do this by having a bag of methods (simple to complex) and applying one by one to the pair of points until we get a connecting path. An example of a simple method is that of trying a straight line segment path between the points in configuration space. This simple method has been successfully used by Kavraki et.al.[6] in their probabilistic roadmap approach, for connecting closely spaced points in CS. An example of a complex method is to parametrize the path between the given points by splines and use a nonlinear optimization strategy to find the spline coefficients that yield a feasible path between them. As extra connections get added, if all EDD components get connected, we stop. On the other hand, if, even after all pairs of points have been tried for connection using all methods, and still there are disconnected EDD components, then we conclude that CS also possibly has disconnected components.

Let x_1 and x_2 be two closely spaced points, lying on different components, EDD1 and EDD2, respectively. We have tried an interesting scheme in which we include an extra artificial distance function in such a way that when the modified EDD is constructed, there is a good chance that EDD1 and EDD2 get connected. Specifically, let the clearances at x_1 and x_2 be c_1 and c_2 respectively, i.e., $c_1 = d_i(x_1)$, $i \in I(x_1)$ and $c_2 = d_i(x_2)$, $i \in I(x_2)$. A pseudo obstacle O_p is placed at location $x_p = 0.5(x_1 + x_2)$ in the configuration space. Note that O_p is a point obstacle in the Configuration Space and has no world space equivalence. The artificial distance function, d_p, is associated with O_p. In order to connect EDD1 and EDD2, $d_p(x)$ has to affect the EDD passing through x_1 and x_2 such that new EDD nodes are created which may help connect EDD1

and EDD2. This is achieved by choosing $d_p(x)$ such that $d_p(x_1) < c_1$ and $d_p(x_2) < c_2$. On the other hand, we want to limit the influence of $d_p(x)$ so that other parts of the EDD that are far away from x_1 and x_2 are not affected. This is achieved by making $d_p(x)$ as large as possible when x is far away from x_p. We do so by selecting the radius of the sphere of influence of O_p as $R = \omega\|x_1 - x_p\|$ and choose $\omega = 1.1$. The distance function associated with O_p takes the form

$$d_p(x) = \begin{cases} -\lambda \log(1 - \|x - x_p\|^2/R^2) & \text{if } \|x - x_p\| \leq \delta R \\ -\lambda \log(0.00975) & \text{if } \|x - x_p\| > \delta R \end{cases}$$

where $\delta = 0.995$ and, λ is chosen just small enough such that $d_p(x_1) < c_1$ and $d_p(x_2) < c_2$. This new distance function $d_p(x)$ is added to the list of N inequalities in formulation (2). The effect of this new function can be seen from Figures 1-3. Figures 1 and 2 show the world space and configuration space for the 2-link manipulator. Figure 2 also shows that the EDD has 3 disconnected components. When 2 pseudo obstacles and the corresponding $d_p(x)$ are included, the modified EDD has only one connected component as shown in Figure 3 (the locations of the pseudo obstacles are marked by a "+").

There is yet another interesting way of connecting different components of EDD. We have mentioned that EDD is representation dependent. To see this, consider the piano mover's problem of the long ellipse shown in Figure 4, moving with translation and rotation in an environment with 9 obstacles (labeled 0-8). The EDD generated using this obstacle field has 6 disconnected components. However, when obstacle 4 is broken into two equal halves and represented as the union of two obstacles, and the same is also done for obstacle 8, it turns out that the EDD has only 1 connected component! For lack of space we do not give the full details here.

Figure 5 shows another piano-mover's problem where the moving object is a "T"-shaped object. The starting and ending configurations are also shown in the Figure. This example is a more difficult problem than that given in Figure 4. However, the standard EDD construction yields a single connected component. Figure 5 also shows details of a motion executed using the EDD.

4 Conclusion

In this paper we have introduced EDD, a new heuristic roadmap method for path planning. The initial results that we have reported here indicate that it is a promising practical method.

References

[1] J.F. Canny, "A Voronoi method for the piano mover's problem", *IEEE Conference on Robotics and Automation*, St. Louis, 1985.

[2] J.F. Canny, *The complexity of robot motion planning*, MIT Press, Cambridge, MA, 1988.

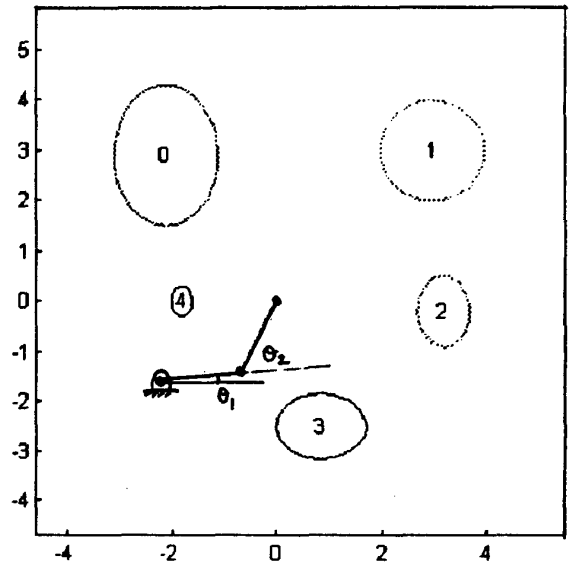

Figure 1: World space of two link manipulator with a pivoted base.

[3] J.F. Canny and M. Lin, "An opportunistic global path planner", *Algorithmica*, Vol. 10, pp. 102-120, 1993.

[4] A. Dattasharma and S.S. Keerthi, "An augmented Voronoi roadmap for 3D translational motion planning for a convex polyhedron moving amidst convex polyhedral obstacles", *Theoretical Computer Science*, Vol. 140, pp. 205-230, 1995.

[5] E.G. Gilbert, D.W. Johnson and S.S. Keerthi, "A fast procedure for computing the distance between complex objects in three dimensional space", *IEEE Journal of Robotics and Automation*, Vol. 4, pp.193-203, 1988.

[6] L. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars, "Probabilistic roadmaps or path planning in high dimensional configuration spaces", *IEEE Transactions on Robotics and Automation*, Vol. 12, 1996, pp. 566-580.

[7] S.S. Keerthi, N.K. Sancheti and A. Dattasharma, "Transversality theorem: A useful tool for establishing genericity", *IEEE Conference on Decision and Control*, 1992.

[8] S.S. Keerthi and M.S. Phatak, "Gain optimization under control structure and stability region constraints", *International Journal of Control*, Vol. 63, pp. 849-864, 1996.

[9] J.-C. Latombe, *Robot motion planning*, Kluwer Academic Publishers, Boston, MA, 1991.

[10] D. Leven and M. Sharir, "Planning a purely translational motion for a convex object in two dimensional space using generalized Voronoi diagrams", *Discrete and Computational Geometry*, Vol. 2, pp. 9-31, 1987.

[11] C.J. Ong and E.G. Gilbert, "Growth distances - new measures for object separation and penetration", *IEEE Transactions on Robotics and Automation*, Vol. 12, pp. 888-903, 1996.

[12] E. Rimon and J. Canny, "Construction of C-Space roadmaps from local sensory data: what should the sensors look for?" *IEEE Conference on Robotics and Automation*, pp. 117-123, 1994, San Diego.

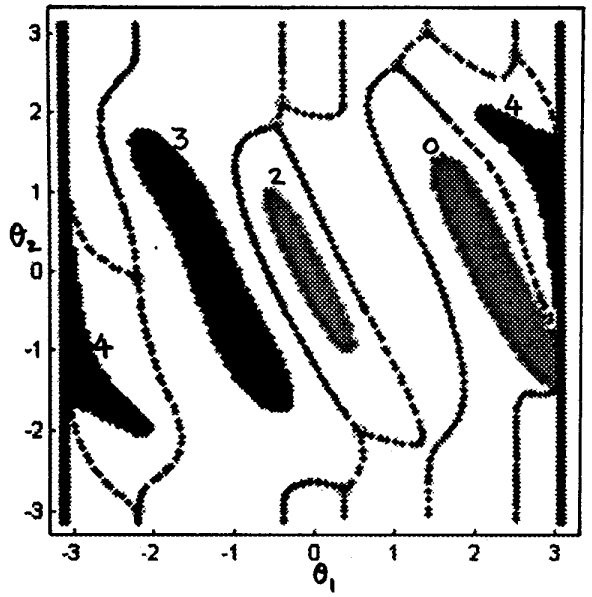

Figure 2: C space of two link manipulator. EDD has three components.

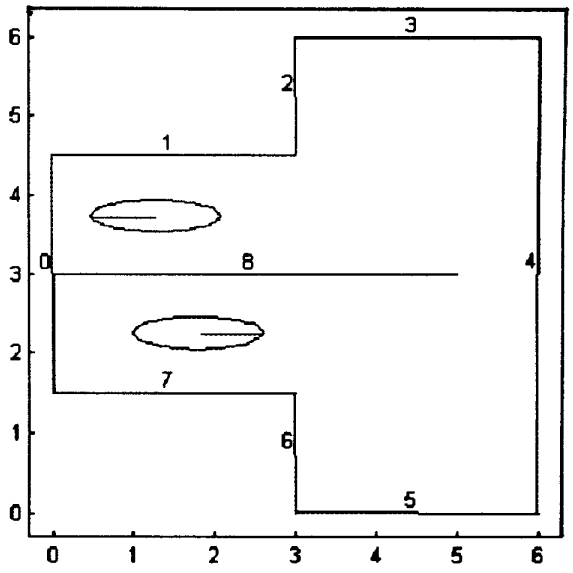

Figure 4: World space for an ellipse moving with translation and rotation.

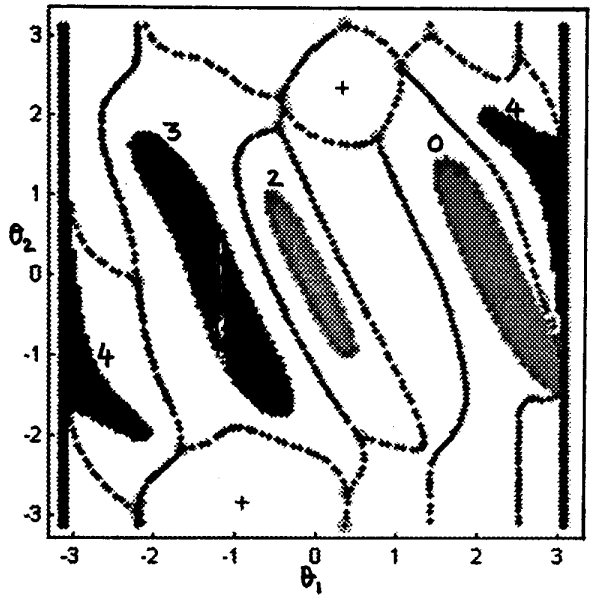

Figure 3: C space of two link manipulator with two artificial distance functions included at points shown with a +. EDD becomes connected.

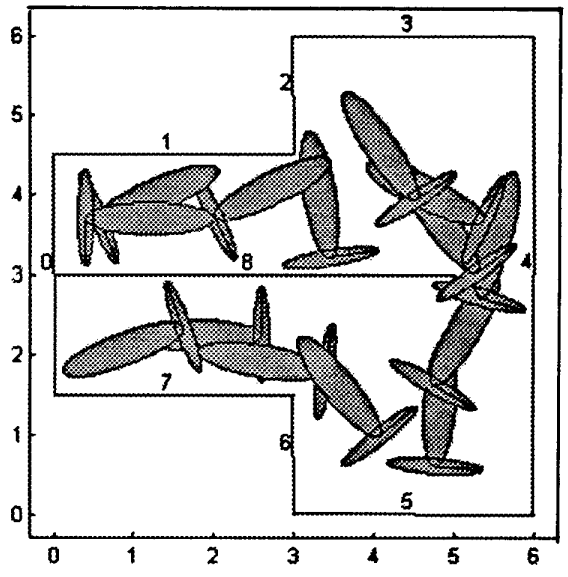

Figure 5: Example of a motion of a T-shaped object moving with translation and rotation, obtained using EDD.