An Opportunistic Global Path Planner

John F. Canny* and **Ming C. Lin†**
University of California, Berkeley
Berkeley, CA 94720

Abstract

In this paper we describe a robot path planning algorithm that constructs a global skeleton of free-space by incremental local methods. The curves of the skeleton are the loci of maxima of an artificial potential field that is directly proportional to distance of the robot from obstacles. Our method has the advantage of fast convergence of local methods in uncluttered environments, but it also has a deterministic and efficient method of escaping local extremal points of the potential function. We first describe a general algorithm, for configuration spaces of any dimension, and then describe instantiations of the algorithm for robots with two and three degrees of freedom.

1 Introduction

There have been two major approaches to motion planning for manipulators, (i) local methods, such as artificial potential field methods [1], which are usually fast but are not guaranteed to find a path, and (ii) global methods, like the first Roadmap Algorithm [2], which is guaranteed to find a path but may spend long time doing it. In this paper we present an algorithm which has characteristics of both. Our method is an incremental construction of a skeleton of free-space. Like the potential field methods, the curves of this skeleton locally optimize a certain potential function that varies with distance from obstacles. Like the Roadmap Algorithm, the skeleton, computed incrementally, is eventually guaranteed to contain a path between two configurations if one exists. The size of the skeleton in the worst case, is comparable with the worst-case size of the roadmap.

Unlike the local methods, our algorithm never gets trapped in local extremal points. Unlike the Roadmap Algorithm, our incremental algorithm can take advantage of a non-worst-case environment. The complexity of the roadmap comes from the need to take recursive slices through configuration space. In our incremental algorithm, slices are only taken when an initial search fails and there is a "bridge" through free space linking two "channels". The conditions for a bridge are quite strict, and can be locally checked before a slice is taken. We expect few slices to be required in typical environments.

In fact, we can make a stronger statement about completeness of the algorithm. The skeleton that the algorithm computes eventually contains paths that are homotopic to all paths in free space. Thus, once we have computed slices through all the bridges, we have a complete description of free-space for the purposes of path planning. Of course, if we only want to find a path joining two given points, we stop the algorithm as soon as it has found a path.

The tracing of individual skeleton curves is a simple enough task that we expect that it could be done in real time on the robot's control hardware, as in other artificial potential field algorithms. However, since the robot may have to backtrack to pass across a bridge, it does not seem worthwhile to do this during the search.

For those readers already familiar with the Roadmap Algorithm, the following description may help with understanding of the new method: If the configuration space is $\Re^k$, then we can construct a surface in $\Re^{k+1}$ which is the graph of the potential function, i.e. if $P(x_1, \ldots, x_k)$ is the potential, the surface is the set of all points of the form $(x_1, \ldots, x_k, P(x_1, \ldots, x_k))$. The skeleton we define here is a subset of a roadmap (in the sense of [2]) of this surface.

This work builds on a considerable volume of work in both global motion planning methods [2] [3], [4], [5], and local planners, [1]. Our method shares common theme with the work of Barraquand and Latombe [6] in that it attempts to use a local potential field planner for speed with some procedure for escaping local maxima. But whereas Barraquand and Latombe's method is a local method made global, we have taken a global method (the Roadmap Algorithm) and found a local opportunistic way to compute it.

Although our starting point was completely different, there are some other similarities with [6]. Our "freeways" resemble the valleys intuitively described in [6]. In fact our freeways are a subset of the valleys described in [6]. But the main difference between our method and the method in [6] is that we have a guaranteed (and reasonably efficient) method of escaping local potential extremal points and that our potential function is computed in the configuration space.

The paper is organized as follows: Section 2 contains a simple and general description of roadmaps. The description deliberately ignores details of things like the distance function used, because the algorithm can work with almost any func-

*Supported by a David and Lucile Packard Foundation Fellowship and by NSF Presidential Young Investigator Grant number IRI-8958577
†Supported by the David and Lucile Packard Foundation

tion. Section 3 gives some particulars of the application of artificial potential fields. Section 4 describes our incremental algorithm, first for robots with two degrees of freedom, then for three degrees of freedom.

2 A General Roadmap Algorithm

We denote the space of all configurations of the robot as CS. For example, for a rotary joint robot with k joints, the configuration space CS is $\Re^k$, the set of all joint angle tuples $(\theta_1, \ldots, \theta_k)$. The set of configurations where the robot overlaps some obstacle is the configuration space obstacle CO, and the complement of CO is the set of free (non-overlapping) configurations FP. As described in [2], FP is bounded by algebraic surfaces in the parameters t_i after the standard substitution $t_i = \tan(\frac{\theta_i}{2})$. This result is needed for the complexity bounds in [2] but we will not need it here.

A roadmap is a one-dimensional subset of FP that is guaranteed to be connected within each connected component of FP. Roadmaps are described in some detail in [2] where it is shown that they can be computed in time $O(n^k \log n)$ for a robot with k degrees of freedom, and where free space is defined by n polynomial constraints (there is also an algebraic component of the complexity which we ignore here). But n^k may still be too large for many applications, and in many cases the free space is much simpler than its worst case complexity, which is $O(n^k)$. We would like to exploit this simplicity to the maximum. The results of [6] suggest that in practice free space is usually much simpler than the worst case bounds. What we will describe is a method aimed at getting a maximal description of the connectivity of a particular free space. The original description of roadmaps is quite technical and intricate. In this paper, we give a less formal and hopefully more intuitive description. We will not give proofs, but the reader who is interested should find that generalizing from the proofs of [2] is straightforward.

2.1 Definitions

The general roadmap is computed recursively on dimension in $CS = \Re^k$. We assume that we have an algorithm that correctly constructs roadmaps of $(k-1)$-dimensional planar slices through $\Re^k$. For concreteness, suppose CS has coordinates $x_1, \ldots, x_k$. A slice $CS|_v$ is a slice by the plane $x_1 = v$. Similarly, slicing FP with the same plane gives a set denoted $FP|_v$. The algorithm is based on the key notion of a channel which we define next:

> A *channel-slice* of free space FP is a connected component of some slice $FP|_v$.

The term channel-slice is used because these sets are precursors to channels. To construct a channel from channel slices, we vary v over some interval. As we do this, for most values of v, all that happens is that the connected components of $FP|_v$ change shape continuously. As v increases, there are however a finite number of values of v, called *critical values*, at which there is some topological change. Some events are not significant for us, such as where the topology of a component of the cross-section changes, but there are four important events:

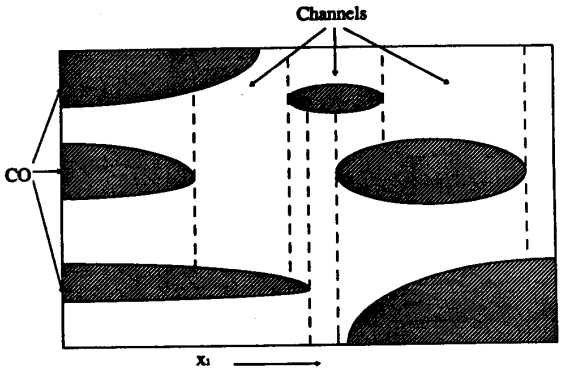

Figure 1: A schematized 2-d configuration space and the partition of free space into x_1-channels.

As v increases a connected component of $FP|_v$ may appear or disappear, or several components may join, or a single component may split into several. The points where joins or splits occur are called *interesting critical points*. We define a channel as a maximal connected union of cross-sections that contains no interesting critical points. We use the notation $FP|_{(a,b)}$ to mean the subset of FP where $x_1 \in (a,b) \subset \Re$.

A *channel* through FP is connected component of $FP_{(a,b)}$ containing no splits or joins, and (maximality) which is not contained in a connected component of $FP_{(c,d)}$ containing no splits or joins, for (c,d) a proper superset of (a,b). See figure 1 for an example of channels. The property of no splits or joins can be stated in another way. A maximal connected set $C|_{(a,b)} \subset FP|_{(a,b)}$ is a channel if every subset $C|_{(e,f)}$ is connected for $(e,f) \subset (a,b)$.

2.2 General Roadmaps

Now to the heart of the method. A roadmap has two components:

(i) Freeways (called silhouette curves in [2]) and

(ii) Bridges (called linking curves in [2]).

A freeway is a connected one-dimensional subset of a channel that forms a backbone for the channel. The key properties of a freeway are that it should span the channel, and be continuable into adjacent channels. A freeway *spans* a channel if its range of x_1 values is the same as the channels, i.e. a freeway for the channel $C|_{(a,b)}$ must have points with all x_1 coordinates in the range (a,b). A freeway is *continuable* if it meets another freeway at its endpoints. i.e. if $C|_{(a,b)}$ and $C'|_{(b,c)}$ are two adjacent channels, the b endpoint of a freeway of $C|_{(a,b)}$ should meet an endpoint of a freeway of $C|_{(b,c)}$. (Technically, since the intervals are open, the word "endpoint" should be replaced by "limit point")

In general, when a specific method of computing freeway curves is chosen, there may be several freeways within one channel. For example, in the rest of this report, freeways are defined using artificial potential functions which are directly proportional to distance from obstacles. In this case each

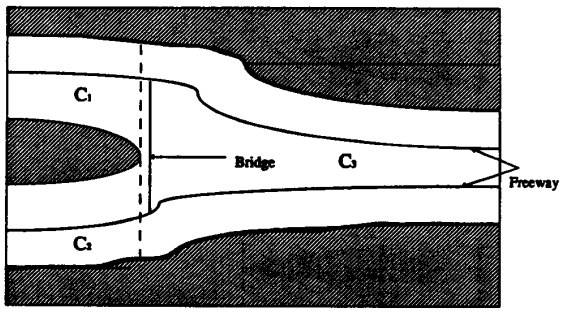

Figure 2: Two channels C_1 and C_2 joining the channel C_3, and a bridge curve in C_3.

freeway is the locus of local maxima in potential within slices $FP|_v$ of FP as v varies. This locus itself may have some critical points, but as we shall see, the freeway curves can be extended easily past them. Since there may be several local potential maxima within a slice, we may have several disjoint freeway curves within a single channel, but with our incremental roadmap construction, this is perfectly OK.

Now to bridges. A bridge is a one-dimensional set which links freeways from channels that have just joined, or are about to split (as v increases). Suppose two channels C_1 and C_2 have joined into a single channel C_3, as shown in figure 2. We know that the freeways of C_1 and C_2 will continue into two freeway curves in C_3. These freeways within C_3 are not guaranteed to connect. However, we do know that by definition C_3 is connected in every slice $x_1 = v$, so we simply call the Roadmap Algorithm recursively, and ask it to construct a roadmap of any non-empty slice $C_3|_v$ of C_3. By our inductive hypothesis, this roadmap is connected. We can also easily arrange things so that this roadmap contains points on both the freeways we are trying to connect. This guarantees that whenever two channels join, their freeways do also. Once we can show that whenever channels meet, their freeways do also (via bridges), we have shown that the roadmap, which is the union of freeways and bridges, is connected. The proof of this very intuitive result is a simple inductive argument on the (finite number of) channels, given in [2].

The basic structure of the general Roadmap Algorithm follows:

1. Start tracing a freeway curve from the start configuration, and also from the goal.
2. If the curves leading from the start and goal are not connected, enumerate a split or join point, and add a bridge curve "near" the split or join (x_1-coordinate of the slice slightly greater than that of the joint point for a join, slightly less for a split).
3. Find all the points on the bridge curve that lie on other freeways, and trace from these freeways. Go to step (2).

The algorithm terminates at step (2) when either the start and goal are connected, in which case the algorithm signals success and returns a connecting path, or if it runs out of split and join points, in which case there is no path connecting the start and goal. This description is quite abstract, but in later sections we will give detailed examples of the approach in two- and three-dimensional configuration spaces.

Two things distinguish our new algorithm from the previous Roadmap Algorithm. One is that the freeways do not necessarily lie near the boundary of free space as they did in [2]. In our present implementation we are in fact using maximum clearance freeways. But the most important difference is that we now only enumerate *true* split or join points. With a robot with k degrees of freedom and an environment of complexity n, it can be shown that there are at most $O(n^{(k-1)})$ potential split or join points. But many experiments with implementated planners in recent years have shown that the number of true splits or joins in typical configuration spaces is much lower. In our new algorithm, we can make a purely local test on a potential split or join point to see if it is really qualified. The vast majority of candidates will not be, so we expect far fewer than $O(n^{(k-1)})$ bridges to be required.

We also plan to experiment with randomly chosen slice values, alternating with slices taken at true split or join points. The rationale for this is that in practice the "range" of slice values over which a bridge joins two freeways is typically quite large. There is good probability of finding a value in this range by using random values. Occasionally there will be an wide range of slices values for a particular bridge, but many irrelevant split and join points may be enumerated with values outside this range. To make sure we don't make such easy problems harder than they should be, our implementation alternates slices taken near true split and join points with slices taken at random x_1 values.

3 The Basic Approach Using an Artificial Potential Field

The idea of our approach is to construct an potential field to repel the point robot in configuration space away from the obstacles. Given a goal position and a description of its environment, a manipulator will move along a "maximum potential" path in an "artificial potential field". The position to be reached represents a critical point that will be linked by the *bridge* to the nearest maximum, and the obstacles represent repulsive surfaces for the manipulator parts.

Let CO denote the obstacles, and x the position in $\Re^n$. The artificial potential field can be described by:

$$U_{art}(x) = U_O(x). \tag{1}$$

The potential field $U_O(x)$ induces an artificial repulsion from the surface of the obstacles. $U_0(x)$ should be defined as a non-negative continuous and differentiable function whose value tends to zero as the end-effector approaches the obstacle's surface. One of the classical analytical potential field is the Euclidean distance function.

Using the shortest distance to an obstacle O, we have proposed the following potential field $U_O(x)$:

$$U_O(x) = \min_i \eta(D_i(O_i, x)) \quad (2)$$

where $D_i(O_i, x)$ is the shortest Euclidean distance between an obstacle O_i and the point x, and η is an adjustment constant. $D_i(O_i, x)$ is obtained by local methods for fast computation of the distance functions. For further details on the computation of distance functions used to construct potential field, please refer to [7].

With this scheme, a manipulator moves in such a way to maximize the artificial potential field $U_{art}(x)$. But, the resulting potential in configuration space may have many local maxima, especially in the obstacle concavities. Thus, it is necessary to design an algorithm to escape these local maxima and to build some linking curves connecting these local maxima until the goal is attained.

In the next section, we will detail the description of our algorithm to guide a robot toward the desired goal and some heuristic techniques to escape local maxima in an "artificial potential field".

4 Description of the New Algorithm

The algorithm will take geometric description of the obstacles in semi-algebraic representations, the initial and goal position as inputs, and output a path between the initial and goal position in robot's configuration space, if such a path exists.

The idea is to construct an potential field function in CS x R to repel the point robot away from the obstacles and toward the desired goal position. This function defines a surface constructed by the Roadmap Algorithm: fix the *v-axis*, then follow the extremal points in *u-axis* as the value of v varies. But, this new algorithm differs from the Roadmap Algorithm[2] in following respects:

- It doesn't always construct the entire skeleton or roadmap
- In this algorithm, $v = x$, where x is one of the CS coordinates while $u = R$, where R is always the height of the potential function in 2-d. Yet, in the Roadmap Algorithm, $v = x$ and $u = y$ where x and y are *both* CS coordinates.
- The Roadmap algorithm fixes one CS coordinate, say x, and follows extremal points (maxima, minima and saddles) in y, to generate the silhouette curves in a two-dimensional workspace. On the other hand, the new algorithm fixes x, and follows only *maxima* in R.

4.1 Two-Dimensional Workspace

Starting from the initial position $s \in CS$, we first fix one of the axes of CS and then the x coordinate of a slice to be the x coordinate of s. Then we search this slice to find the nearest local maximum. (This local maximum is a freeway point.) Next, we build a *bridge* between the point s and this local maximum. At the same time, we begin tracing a freeway

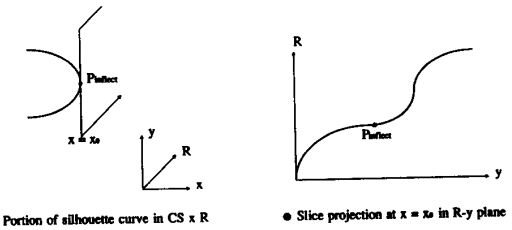

Figure 3: A pictorial example of an inflection point in CS x R vs. its view in R x y at the slice $x = x_0$

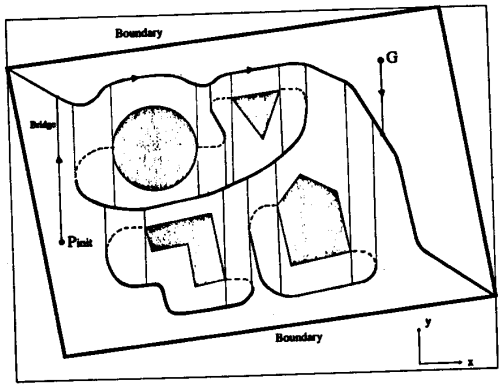

Figure 4: An example of the algorithm in the 2-d workspace

curve from the goal. If the goal is not on the maximum contour of potential field, then we must build a *bridge* to link it to the nearest local maximum. Afterwards, we trace the locus of this local maximum as x varies until we reach an endpoint. If the current position p_{loc} on the curve is the goal G, then we can terminate the procedure. Otherwise, we must verify whether p_{loc} is a "dead-end" or an inflection point of the slice $x = x_0$. (See Fig. 3.) If p_{loc} is a point of inflection, then we can continue the curve by taking a slice at the inflection point and following along the gradient direction *near* the inflection point. This search necessarily takes us to another local maximum. Fig. 4 demonstrates how the algorithm works in 2-d CS. This diagram is a projection of a constructed potential field in CS x R onto the x-y plane of the 2-d CS. The shaded area is the CO in the configuration space. The solid curves represent the contour of maximum potential, while the dashed curves represent the minima. Furthermore, the path generated by our planner is indicated by arrows. In addition, the vertical lines symbolize channel slices through the interesting critical points. When this procedure has been taken to its conclusion, both endpoints of the freeway will terminate at dead-ends. At this point it is necessary to take a recursive

slice at some value of x. Our planner generates several random x-values for slices (at a uniformly spaced distribution along the span of the freeway), interweaving them with an enumeration of all the interesting critical points. If after a specified number of random values, our planner fails to find a connecting path to a nearby local maximum, then it will take a slice through an interesting critical point. Each slice gives a bridge which is itself the recursively computed roadmap of the slice. We call this procedure recursively until we reach the goal position G or have enumerated all the interesting critical points.

The algorithm is described schematically below:

- Algorithm

```
Procedure FindGoal (Environment, p_init, G)

    if (p_init ≠ G)
       then Explore(p_init)
       else return(FoundGoal);
    even := false;
    While ( CritPtRemain and NotOnSkeleton(G) ) do
       if (even)
          then x := Random (x-range)
          else x := x-coord(next-crit-pt());
       TakeSlice(x);
       even := not even;
    end(while);
End(FindGoal);

Function Explore(p)
% Trace out a curve from p

    q := search-up&down(p);
    % To move up & down only in y, using gradient near p
    if new(q) then
    % new() checks if q is already on the curve
       begin(if)
       <e1,e2> := trace(q);
       % trace out the curve from q, output two end points
       if inflection(e1) then Explore(e1);
       if inflection(e2) then Explore(e2);
       % inflection(p) checks if p is an inflection point
       end(if);
End(Explore);

Function TakeSlice(x-coordinate(p))
% This function generates all points on the slice and explore
% all the maxima on the slice.

    old-pt := find-pt(x-coordinate);
    % find-pt() find all the points on the x-coordinate.
    % It moves up&down until reaches another maximum.
    new-pt := null;
    For (each pt in the old-pt) do
       <up,down> := search-up&down(pt);
       % <up,down> is a pair of points of 0,1,or2 pts
       new-pt := new-pt + <up,down>;
    For (each pt in the new-pt) do
       Explore(pt);
End(TakeSlice);
```

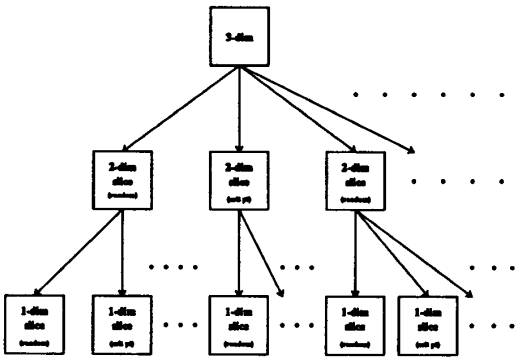

Figure 5: Search Tree for Taking Slices in Three-D

4.2 Three-Dimensional Workspace

For a three-dimensional workspace, we can compute the path in a similar fashion. Starting from the initial position p_{init}, we first fix one axis. Without loss of generality, we find the local maxima on the X-Y plane as the values of Z vary. A "maximum potential" path following the gradient direction of potential field can be traced out in the three-dimensional space. Then, we repeat the verifying process as in the two-dimensional case. If the current position is an inflection point, we can always reach another maximum by following the direction of potential gradient to another local maximum. Once we reach a maximum, however, we must find another nearby maximum in the 3-dimensional space and then build a bridge curve connecting the two maxima.

The search for a local maximum is analogous to the two-dimensional case. First, we take a random two-dimensional slice $FP|_z$ normal to the Z-axis and search for local maxima on $FP|_z$ using the algorithm of the last section. This algorithm in turn may take one-dimensional slices, alternating between random search and enumeration of critical points. But the random slice $FP|_z$ may not always contain a local maximum or else it may be quite difficult to find a local maximum on this slice. Therefore, we interweave the execution of the two algorithms. After we take a certain number of one-dimensional slices within $FP|_{z_1}$, we either take new two-dimensional slice $FP|_{z_2}$ or explore some previous slice $FP|_{z_0}$ further by taking more one-dimensional slices. Hence, the search for local maxima problem becomes more complex in a three-dimensional space, since now we are facing different searching options as shown in the search tree (see Fig. 5). An "optimal" decision will be made at each node on the tree by a heuristic search algorithm that is currently under investigation. In conclusion, for planning a path in 3-dimensional workspace, we apply the algorithm recursively by taking two-dimensional slices and then one-dimensional slices, with a

heuristic search that trades off path optimality and search time.

4.3 Path Optimization

After the solution path is obtained, we plan to smooth it by the classical principles of variational calculus. That is, to solve a classical two points boundary value problem. Basically, we minimize the potential that is a function of both distance and smoothness, or to find an "optimal" path given the algorithm.

5 Complexity Bound

Since our planner probably does not need to explore all the critical points, this bound can be reduced by finding only those *interesting* critical points where adding a bridge helps to reach the goal. If n is the number of obstacle features (faces, edges, vertices) in the environment and the configuration space is R^k, then the number of "interesting critical points" is at most $O(n^{k-1})$. For instance, in a three-dimensional workspace, for an environment with n obstacles, the number of "interesting critical points" is at most $O(n^2)$ and the number of "interesting critical points" for the 2-D algorithm is at most $O(n)$. (Please refer to Append I for more details.)

6 Summary and Discussion

By following the maxima of a well-designed potential field, and taking slice projections through critical points and at random values, our approach builds incrementally an obstacle-avoiding path to guide a robot toward the desired goal. The techniques proposed in this paper provide the planner with a systematic way to escape from these local maxima that have been a long standing problem with using the potential field approach in robot motion planning.

Our algorithm, computed from local information about the geometry of configuration space, requires no expensive precomputation steps as in most global methods developed thus far. In a two dimensional space, this method is comparable with using a Voronoi Diagram for path planning. In three-dimensional space, however, our method is more efficient than computing hyperbolic surfaces for Voronoi diagram method. In the worst case, it will run at least as fast as the Roadmap Algorithm. But, it should run faster than the Roadmap Algorithm in most of the cases.

The implementation of this algorithm is in progress at the present time and the issue of optimal search is also under investigation.

Appendix I: Geometric Relations between Critical Points and Contact Constraints

Let n be the number of obstacle features in the environment. Free space FP is bordered by $O(n)$ constraint surfaces. Each constraint surface corresponds to an elementary contact, either face-vertex or edge-edge, between a feature of the robot and a feature of the environment. Other types of contact are called non-elementary, and can be viewed as multiple elementary contacts at the same point, e.g. vertex-edge. They correspond to intersections of constraint surfaces in configuration space. There are also $O(n)$ non-elementary contact surfaces in configuration space, and each has co-dimension 2 or more.

In k dimensions, the arrangement of constraint surfaces has $O(n^k)$ intersection points. These intersection points fall into two categories: (a) All the contacts are elementary (b) one or more contacts are non-elementary. When all contacts are elementary, i.e. all contact points are distinct on the object, free space in a neighborhood of the intersection point is the intersection of k half planes (one side of a constraint surface), and forms a cone. This type of intersection point cannot be a split or join point, and does not require a bridge. However, if one or more contacts are non-elementary, then the intersection point is a potential split or join point. But because the $O(n)$ non-elementary contact surfaces have codimension ≥ 2 there are only $O(n^{k-1})$ intersection points of type (b). Interesting critical points may be either intersection points, and we have seen that there are $O(n^{k-1})$ candidates, or they may lie on higher dimensional intersection surfaces, but there are at most $O(n^{k-1})$ intersection surfaces of dimension one or greater. Therefore, the number of interesting critical points is at most $O(n^{k-1})$.

References

[1] O. Khatib. Real-time obstable avoidance for manipulators and mobile robots. *IJRR*, 5(1), 1986.

[2] J. F. Canny. *The Complexity of Robot Motion Planning.* MIT Press, Cambridge, MA, 1988.

[3] T. Lozano-Pérez and M. Wesley. An algorithm for planning collision-free paths among polyhedral obstacles. *Comm. ACM*, 22(10):560–570, 1979.

[4] J. Reif. *Complexity of the Mover's Problem and Generalizations*, chapter 11, pages 267–281. Ablex publishing corp., New Jersey, 1987.

[5] J.T. Schwartz and M. Sharir. *On the 'Piano Movers' Problem, II. General Techniques for Computing Topological Properties of Real Algebraic Manifolds*, chapter 5, pages 154–186. Ablex publishing corp., New Jersey, 1987.

[6] B. Langlois J. Barraquand and J-C. Latombe. Robot motion planning with many degrees of freedom and dynamic constraints. In *Proceedings 5th ISRR*, Tokyo, Japan, 1989.

[7] J. F. Canny and M. C. Lin. Local methods for fast computation of distance functions. In Preparation, 1990. U. C. Berkeley.